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

保守／廃止

μSAP30100-B01
μSAP703000-B01
μSAP705100-B01
μSAP70732-B01

MH/MR/MMR ミドルウエア

対象デバイス

μSAP30100-B01　：VR4100 シリーズ™
μSAP703000-B01：V850 ファミリ™
μSAP705100-B01：V830 ファミリ™
μSAP70732-B01　：V810 ファミリ™

資料番号　U10338JJ4V0UM00（第４版）
発行年月　July 1998　N　CP(K)

保守／廃止

〔メ　モ〕

保守／廃止

目　次　要　約

M7A　94.11

保守／廃止

本版で改訂された主な箇所

箇　所	内　容
全般	μSAP30100-B01（VR4100シリーズ用　MH/MR/MMRミドルウエア）を追加
p. 23	**1.2.2　MH方式**　修正
p. 45	**2.1.2（2）バッファ・アドレスとバッファの残りサイズの取り扱い**　修正
p. 111	**2.3.2（8）MR複合化**に不正なコードの処理の記述追加
p. 147	**図4－12　サンプル・プログラム（圧縮／伸長）のメモリ・マップ（AP70732-B01）：V810ファミリの場合**　修正

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™，VR4100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μSAP30100-B01，μSAP703000-B01, μSAP705100-B01, μSAP70732-B01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

読み方　このマニュアルでは「μSAP30100-B01」という製品名を「AP30100-B01」に，「μSAP703000-B01」という製品名を「AP703000-B01」に，「μSAP705100-B01」という製品名を「AP705100-B01」に，「μSAP70732-B01」という製品名を「AP70732-B01」に置き換えて説明しています。

このマニュアルの読者には，電気，論理回路，マイクロコンピュータおよびC言語に関する一般知識を必要とします。

V800シリーズのハードウエア機能を知りたいとき
　→各製品のユーザーズ・マニュアル　ハードウエア編を参照してください。

V800シリーズの命令機能を知りたいとき
　→各製品のユーザーズ・マニュアル　アーキテクチャ編を参照してください。

VR4100シリーズのハードウエア機能／命令機能を知りたいとき
　→各製品のユーザーズ・マニュアル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
メモリ・マップのアドレス：上部－上位，下部－下位
注　　：本文中に付けた注の説明
注　意　：気を付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…xxxxまたはxxxxB

10進数…xxxx

16進数…xxxxHまたは0x xxxx

２のべき数を示す接頭語（アドレス空間，メモリ容量）：

K（キロ）：$2^{10} = 1024$

M（メガ）：$2^{20} = 1024^2$

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

V810ファミリに関する資料

品名＼資料名		データ・シート	ユーザーズ・マニュアル	
			ハードウエア編	アーキテクチャ編
V810™	μPD70732	U10691J	U10661J	U10082J
V821™	μPD70741	U11678J	U10077J	

V830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	
			ハードウエア編	アーキテクチャ編
V830™	μPD705100	U11483J	U10064J	U12496J
V831™	μPD705101	U12979J	U11483J	

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	
			ハードウエア編	アーキテクチャ編
V852™	μPD703002	U11826J	U10038J	U10243J
	μPD70P3002	U11827J		
V853™	μPD703003	U12261J	U10913J	
	μPD703003A, 703004A, 703025A	U13188J		
	μPD70F3003	U12036J		
	μPD70F3003A, 70F3025A	U13189J		
V854™	μPD70F3008	U12756J	U11969J	
	μPD70F3008Y	U12755J		

保守／廃止

VR4100シリーズに関する資料

品名＼資料名		データ・シート	ユーザーズ・マニュアル	アプリケーション・ノート
VR4100™	μPD30100	U10428J	U10050J	プログラミング・ガイド（U10710J）
VR4102™	μPD30102	U12543J	U12739J	
VR4111™	μPD30111	U13211J	U13137J	

V810ファミリの開発ツールに関する資料（ユーザーズ・マニュアル）

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

V830ファミリの開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号
IE-705100-MC-EM1　（V830用インサーキット・エミュレータ）		U11869J
IE-70000-MC-NW　（V831用インサーキット・エミュレータ）		U12476J
CA830　（Cコンパイラ）	操作編（UNIXベース）	U11013J
	操作編（Windowsベース）	U11068J
	アセンブリ言語編	U11014J
	C言語編	U11010J
	プロジェクト・マネージャ編	U11991J
ID830　（Cソース・ディバッガ）	操作編（UNIXベース）	U12024J
	操作編（Windowsベース）	U12206J
	インストレーション編（UNIXベース）	U12023J
RX830　（リアルタイムOS）	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ファミリの開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号
IE-703002-MC（V852, V853, V854用インサーキット・エミュレータ）		U11595J
IE-703003-MC-EM1（V853用周辺I/Oボード）		U11596J
IE-703008-MC-EM1（V854用周辺I/Oボード）		U12420J
CA850　（Cコンパイラ・パッケージ）	操作編　UNIXベース	U12839J
	操作編　Windowsベース	U12827J
	C言語編	U12840J
	アセンブリ言語編	U10543J
RX850　（リアルタイムOS）	基礎編	U13430J
	テクニカル編	U13431J
	インストレーション編	U13410J
RD850　（タスク・ディバッガ）	Windowsベース	U11158J
AZ850　（システム・パフォーマンス・アナライザ）	Windowsベース	U11181J
ID850　（Cソース・ディバッガ）	操作編　Windowsベース	U11196J

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

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

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

保守／廃止

目　　次

保守／廃止

図　の　目　次（1/2）

保守／廃止

図　の　目　次（2/2）

保守／廃止

表　の　目　次

保守／廃止

〔メ　モ〕

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余すことなく引き出せるようにチューニングされたソフトウエア群で，従来ハードウエアが行っていた処理をソフトウエアで実現したものです。RISC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そろってきたため，ミドルウエアという概念がまさに現実のものとなってきました。

★ NECでは，VR4100シリーズ，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，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。

AP30100-B01, AP703000-B01, AP705100-B01, AP70732-B01は，FAXコーデック機能を提供するミドルウエアです。

１.２　MH/MR/MMR

MH/MR/MMRは，ITU-T（International Telecommunications Union：国際電気通信連合）勧告により定められた２値画情報の符号化（圧縮）方式です。復号化（伸長）は符号化と逆の手順で実現します。ITU-T勧告についての内容は「ITU-T　ホワイト・ブック　G3/G4ファクシミリ関連　Tシリーズ勧告」（財団法人　新日本ITU協会刊）を参照してください。

MH　：Modified Huffman
MR　：Modified Relative element address designate
MMR　：Modified MR

１.２.１　ラン・レングス符号化

図１－１のような絵をスキャナなどで２値画情報として取り込んだ場合，メモリ上には図１－２のようなビット・マップ・データとして取り込まれます。つまり，白の画素は０，黒の画素は１で表されます。

保守／廃止

図1－1　サンプル図

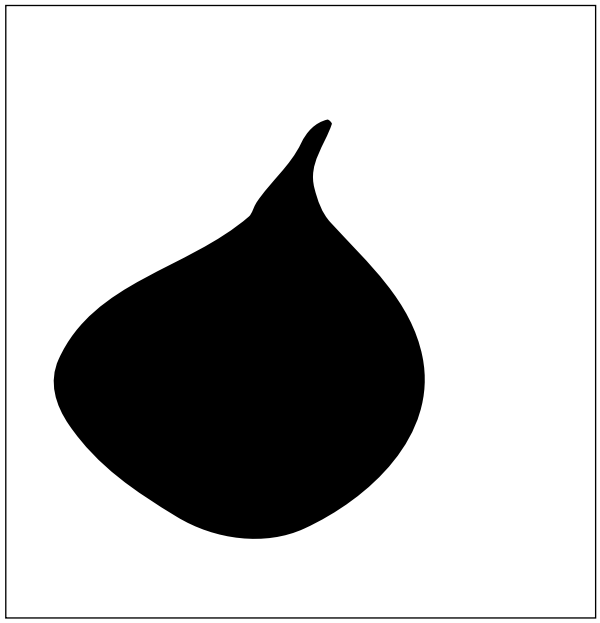
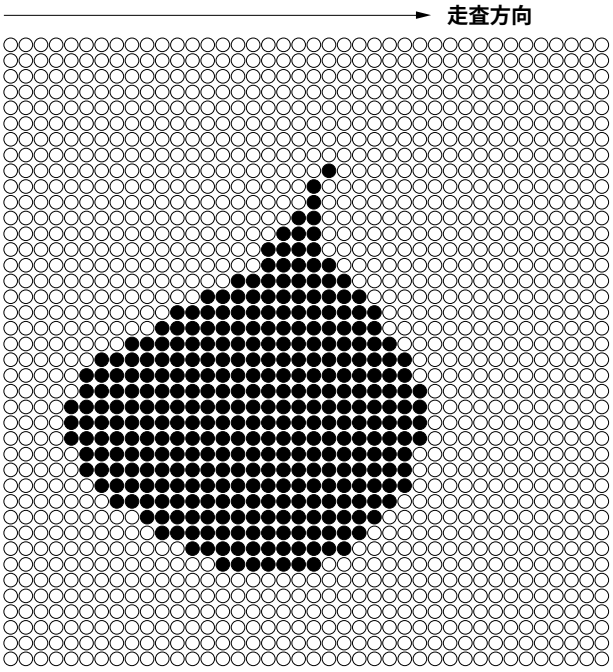


図1－2　ビット・マップ・データ



備考　○：ビット0
　　●：ビット1

これを図１－２の矢印の方向に走査していくと，初めは０のビットが連続していますが，あるところで０→１へ変化する点（変化点）にぶつかります。さらに，走査を進めると今度は１→０への変化点にぶつかります。ラン・レングス符号化とは，ひとつの変化点から次の変化点までの連続する同じ色（白か黒）の画素数（これをラン・レングスといいます）を何らかの方法で符号に置き換えるものです。

たとえば，符号としてラン・レングスそのものを使えば，図１－２のビット・マップ・データは図１－３のような数値列に置き換わります。元のデータ量は40×40＝1600ビット＝200バイトですが，符号化後は数十バイトに圧縮できます。スキャナの解像度を上げれば，この差はさらに大きくなります。

図１－３　ラン・レングス符号列

341 1 38 1 39 1 38 2 37 3 36 4 36 5 37 8 30 11 27 13 25 15 23 18 20 21 18
22 18 23 16 24 16 24 16 24 17 22 18 22 20 19 23 16 25 14 28 11 31 7 259

符号を復号して元の画像を再生するには，走査の開始点の色と画像の横サイズが分かれば符号化の逆の手順によって容易に実現できます。しかも，この再生画像は元の画像と完全に同じものです。

以上がラン・レングス符号化の考え方です。次項より説明するITU-T勧告の符号化方式は，このラン・レングス符号化が基礎となっています。

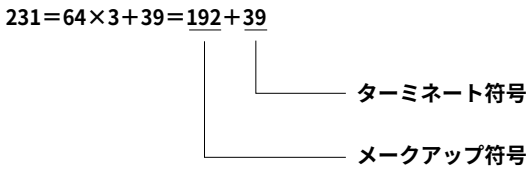
保守／廃止

1.2.2　MH方式

MH方式は一次元符号化方式とも呼ばれます。

この方式では，ラン・レングスを画像データの各水平ラインごとに計数します。各ラン・レングスに対する符号としては，表１－１のものを採用しています。符号にはメークアップ符号とターミネート符号の２種類があります。64個未満のラン・レングスに対してはターミネート符号が割り当てられ，64個以上のラン・レングスについてはメークアップ符号とターミネート符号の組み合わせが割り当てられます（**例１，例２**参照）。

例１．ラン・レングスが231の場合



２．ラン・レングスが128の場合

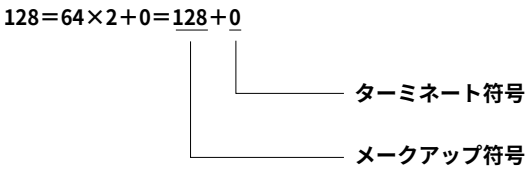


表1－1　ラン・レングス符号

（a）ターミネート符号

ラン・レングス	白ラン用符号語	黒ラン用符号語	ラン・レングス	白ラン用符号語	黒ラン用符号語
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	1000	10	35	00010100	000011010011
4	1011	011	36	00010101	000011010100
5	1100	0011	37	00010110	000011010101
6	1110	0010	38	00010111	000011010110
7	1111	00011	39	00101000	000011010111
8	10011	000101	40	00101001	000001101100
9	10100	000100	41	00101010	000001101101
10	00111	0000100	42	00101011	000011011010
11	01000	0000101	43	00101100	000011011011
12	001000	0000111	44	00101101	000001010100
13	000011	00000100	45	00000100	000001010101
14	110100	00000111	46	00000101	000001010110
15	110101	000011000	47	00001010	000001010111
16	101010	0000010111	48	00001011	000001100100
17	101011	0000011000	49	01010010	000001100101
18	0100111	0000001000	50	01010011	000001010010
19	0001100	00001100111	51	01010100	000001010011
20	0001000	00001101000	52	01010101	000000100100
21	0010111	00001101100	53	00100100	000000110111
22	0000011	00000110111	54	00100101	000000111000
23	0000100	00000101000	55	01011000	000000100111
24	0101000	00000010111	56	01011001	000000101000
25	0101011	00000011000	57	01011010	000001011000
26	0010011	000011001010	58	01011011	000001011001
27	0100100	000011001011	59	01001010	000000101011
28	0011000	000011001100	60	01001011	000000101100
29	00000010	000011001101	61	00110010	000001011010
30	00000011	000001101000	62	00110011	000001100110
31	00011010	000001101001	63	00110100	000001100111

（b）メークアップ符号

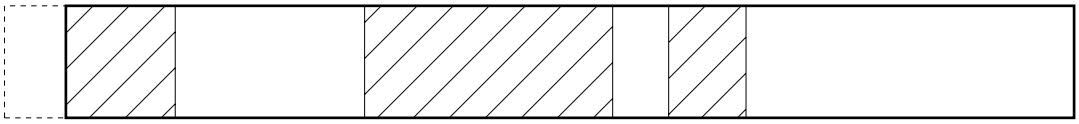
ラン・レングス	白ラン用符号語	黒ラン用符号語	ラン・レングス	符号語（白ラン，黒ラン共通）
64	11011	0000001111	1792	00000001000
128	10010	000011001000	1856	00000001100
192	010111	000011001001	1920	00000001101
256	0110111	000001011011	1984	000000010010
320	00110110	000000110011	2048	000000010011
384	00110111	000000110100	2112	000000010100
448	01100100	000000110101	2176	000000010101
512	01100101	0000001101100	2240	000000010110
576	01101000	0000001101101	2304	000000010111
640	01100111	0000001001010	2368	000000011100
704	011001100	0000001001011	2432	000000011101
768	011001101	0000001001100	2496	000000011110
832	011010010	0000001001101	2560	000000011111
896	011010011	0000001110010		
960	011010100	0000001110011		
1024	011010101	0000001110100		
1088	011010110	0000001110101		
1152	011010111	0000001110110		
1216	011011000	0000001110111		
1280	011011001	0000001010010		
1344	011011010	0000001010011		
1408	011011011	0000001010100		
1472	010011000	0000001010101		
1536	010011001	0000001011010		
1600	010011010	0000001011011		
1664	011000	0000001100100		
1728	010011011	0000001100101		
EOL	000000000001	000000000001		

図１－４に１ライン分の符号化の例を示します。ラインの先頭画素が黒で始まる場合には，その直前に０個の仮想的な白画素があるものとして符号化します。

図１－４　１ライン分の符号化例

画像データ

白0　黒15　白139　黒640　白14　黒18　白902



計1728ビット

符号データ

白0	黒15	白128	白11	黒640	黒0	白14	黒18	白896	白6
00110101	000011000	10010	01000	0000001001010	0000110111	110100	0000001000	011010011	1110

計79ビット

１ページ分の符号化は次の手順で行います。

① 最初に符号表中のEOL（End Of Line）符号を出力します。

② １ラインを符号化します。

③ 必要ならば，最少伝送時間[注]を保証するために複数個の０から成るフィル・ビットを付加します。

④ EOL符号を付加します。

★ ⑤ ②から④を繰り返します。

⑥ １ページの最終ラインの符号（またはそれに続くフィル・ビット）の後ろにRTC（Return To Control）符号を付加してページの終端を示します。ただし，

RTC = EOL×6

とします。

注　最少伝送時間

ファクシミリなどにおいては，受信側の受信許容速度がまちまちです。受信許容速度は主として，受信バッファの大きさやプリント・アウトの速度に依存して決まります。符号データの量は画像データのパターンに依存し，送信速度は符号データの量に依存します。したがって，この受信許容速度が遅いにもかかわらず短いライン符号が次々と送られてきた場合には，１ライン分の画像を出力し終わらないうちに次のライン符号を受信しなければならなくなり，データの取りこぼしが生じることになります。

このため，１ライン分の符号データを受信できる最少の時間（最少伝送時間）が機種ごとに決まっています。送信側が１ライン分の符号データを送信する場合に，この時間を守れないとき

はフィル・ビットを挿入して調整してください。また，送り先の最少伝送時間は符号データの送信を開始する前に，それぞれの機器間で連絡をとることになっています。

この手順で符号化することにより，１ページ分の画像データは図１－５に示すようになります。

図１－５　１ページ分のMH符号

備考　RTC=EOL×6

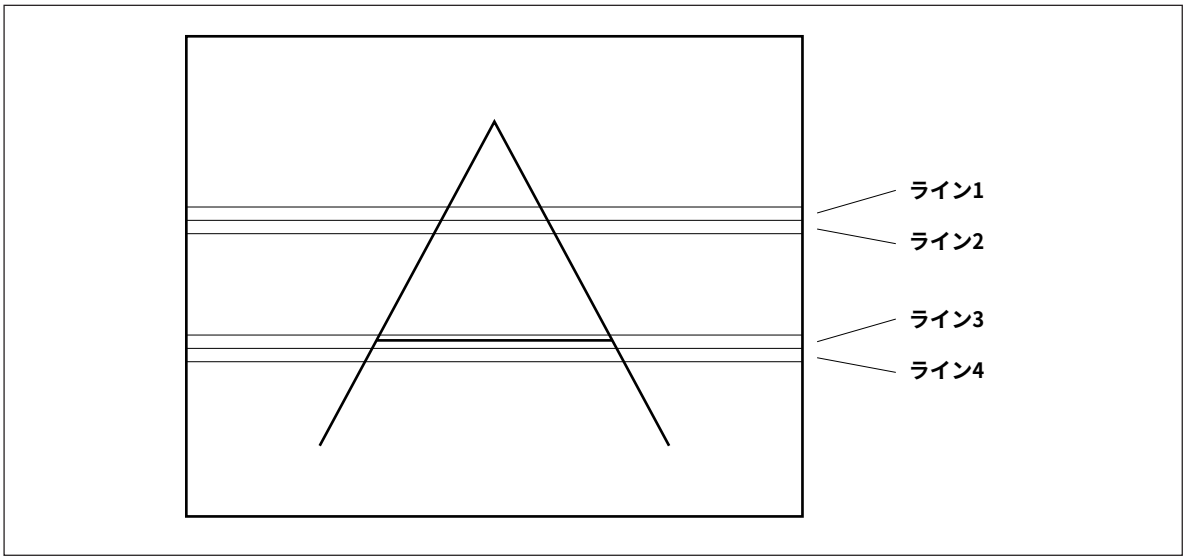
保守／廃止

1.2.3　MR方式

MR方式は，画像データの垂直方向の相関に着目して，符号化するラインを１つ手前のラインと比較しながら符号化を進める方式です。

文字や図形などが描かれた文書画像では基本的に縦，横，斜めの線で画像が構成されています。したがって，図１－６のライン１とライン２のように，あるラインについて特定の画像パターンが現れたならば，その上下のラインについても同じようなパターンが現れる頻度が高いと言えます。同様に，図１－６のライン３とライン４のように，上下のラインでまったく違う画像パターンが現れる頻度も高いと言えます。そこで，このような２つの場合に対して短い符号を割り当てることによって，MH方式以上の圧縮率が期待できます。

図１－６　画像の垂直方向の相関

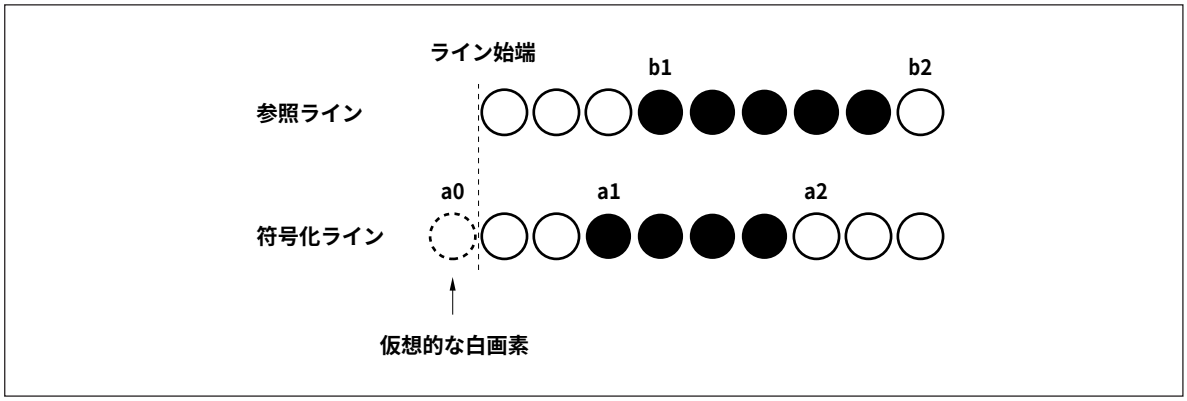


この方式の説明をする前に説明で使用する用語と記号の定義をします。

符号化するラインの１つ手前のラインを参照ラインと呼びます。符号化するライン上と参照ライン上の変化点に対して，図１－７のようにa1，a2，b1，b2のレーベルを付けます。この４個の画素の位置関係から３つのモードを考えて符号化します。a0は基準点と呼ばれる走査を開始する画素です。５個の画素は次のように定義されています。

① a0：各ラインを符号化する際に，ラインの最初の画素の直前に仮想的な白画素があるものとして，ここに置かれます。a0を開始点として１ラインのある範囲の符号化が終了すると，符号化された領域の後ろに新たにa0の位置が決定されます。
② a1：符号化するライン上で，a0の次に現れる変化点に置かれます。
③ a2：符号化するライン上で，a1の次に現れる変化点に置かれます。
④ b1：参照ライン上で，a0の右に現れる，a0の画素の色と反対色の最初の画素に置かれます。
⑤ b2：参照ライン上で，b1の次に現れる変化点に置かれます。

図１－７　符号化の基準となる点



符号化のモードとしては，次の３つがあります。

・パス・モード
・垂直モード
・水平モード

それぞれのモードの詳細は次のとおりです。

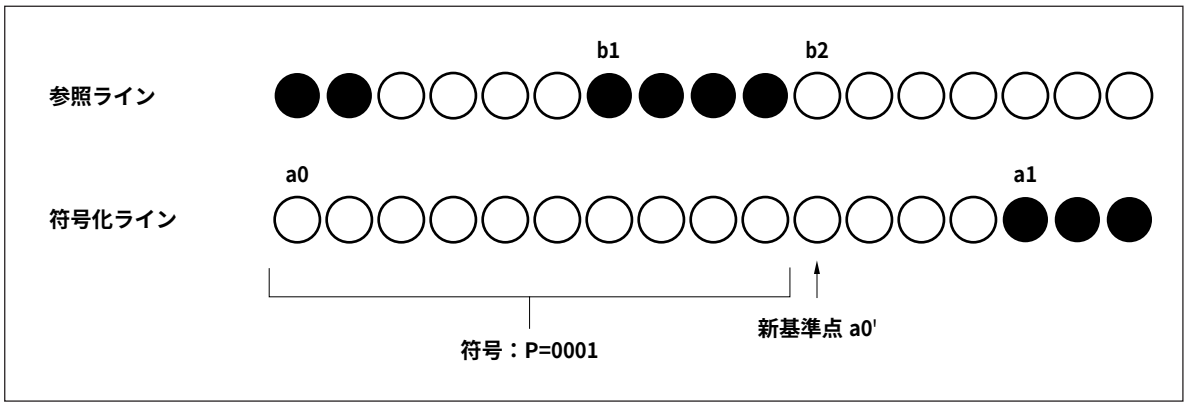
（1）パス・モード

パス・モードは，b2がa1の左側にある場合に対応します。このことは，符号化するライン上でのa0の次の変化点が，b2に対応する位置までは存在しないことを意味します（**図１－８**）。図１－６のライン３とライン４のような組み合わせで起こります。

具体的な符号化は，次の手順で行います。

① 符号としてパス・モード符号P（= 0001）を出力します。
② b2に対応する位置に新基準点a0'を設定します。a0'は以後の符号化において，a0となります。

図１－８　パス・モードの例



保守／廃止

（２）垂直モード

垂直モードは，a1がb1の左右計７画素以内の範囲にある場合に対応します（**図１－９**）。図１－６のライン１とライン２のような組み合わせで起こります。

具体的な符号化は，次の手順で行います。

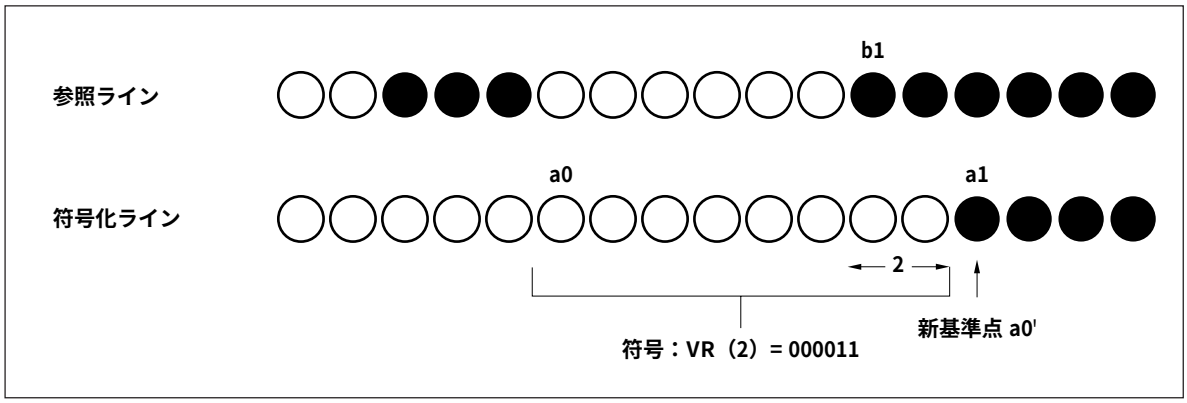
① 符号として，b1に対するa1の相対的な位置（右側か左側）および距離（画素数）に着目して表に従った符号を出力します（**表１－２**）。

② a1の位置に新基準点a0'を設定します。a0'は以後の符号化において，a0となります。

表１－２　垂直モードの符号

種類	a1とb1との位置関係	符号
V（0）	a1がb1の真下にある	1
VL（1）	a1がb1から１画素分左にある	010
VL（2）	a1がb1から２画素分左にある	000010
VL（3）	a1がb1から３画素分左にある	0000010
VR（1）	a1がb1から１画素分右にある	011
VR（2）	a1がb1から２画素分右にある	000011
VR（3）	a1がb1から３画素分右にある	0000011

図１－９　垂直モードの例（a1がb1の右側２画素目にある場合）



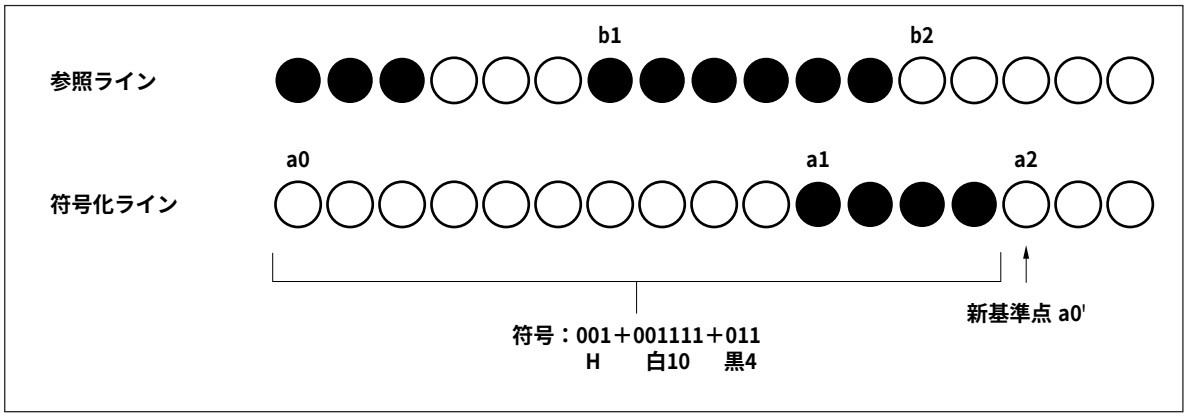
（3）水平モード

パス・モードまたは垂直モード以外の場合，つまりb2がa1の右側にあり，かつa1とb1の距離が３画素を越える場合に対応します（**図１－10**）。

具体的な符号化は，次の手順で行います。

① 符号として，まず水平モード符号H（=001）を出力します。
② a0からa1までのラン・レングスをMH方式の符号表（**表１－１**）に従って符号化します。
③ a1からa2までのラン・レングスをMH方式の符号表（**表１－１**）に従って符号化します。
④ a2の位置に新基準点a0'を設定します。a0'は以後の符号化において，a0となります。

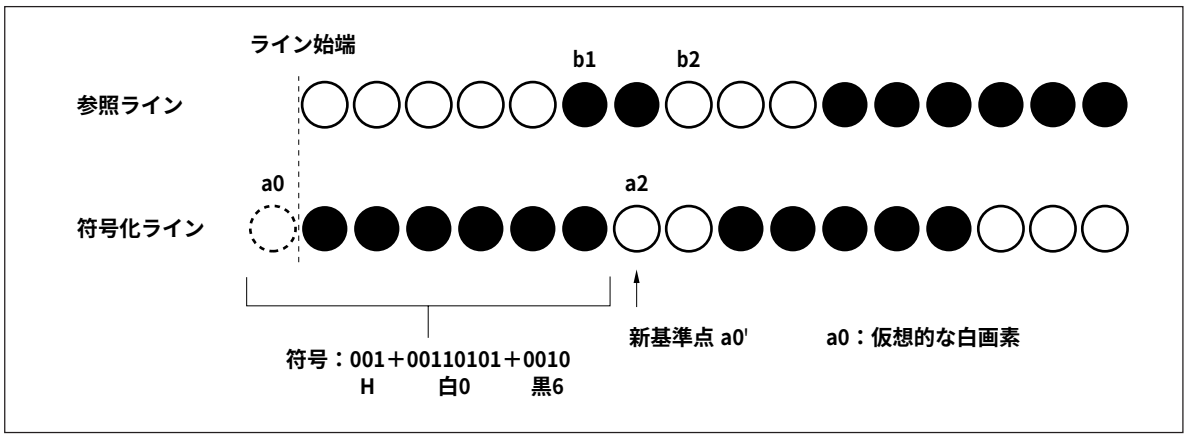
図１－10　水平モードの例



各ラインの始端と終端ではa0-a2，b1-b2の関係が特殊なものとなるため特別な処理が必要です。

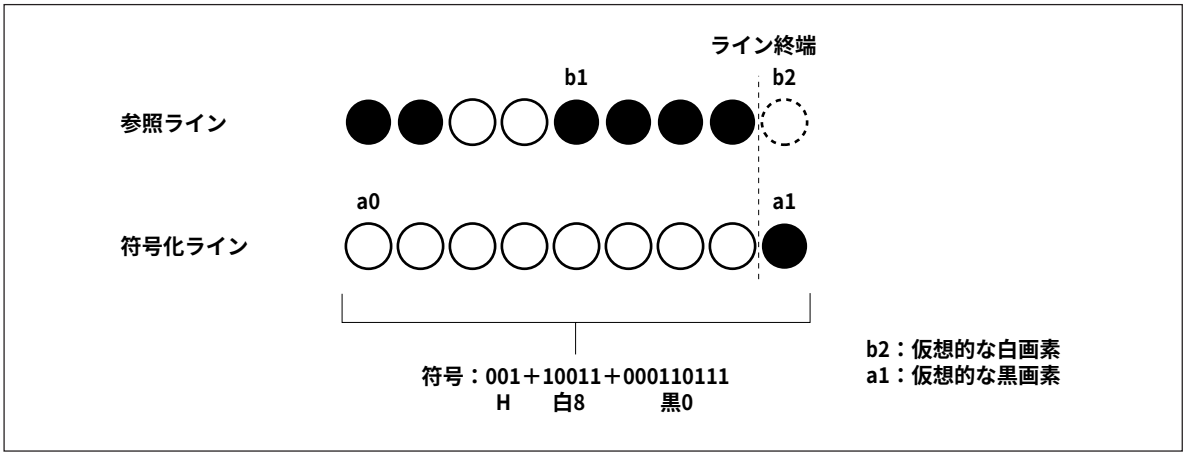
ライン始端では，先頭の画素が黒の場合，ラインの最初の画素の直前に０個の仮想的な白画素があるものとして，a0がここに置かれます。したがって，図１－11のようにラインが黒画素で始まっているような場合，符号はHに続いてラン・レングス０の符号が来ることになります。

図１－11　ライン始端の処理例



ライン終端では，最後の画素の直後にそれと反対色の仮想的な画素が０個続くものとして，仮想画素までを符号化します（**図１－12**）。

図１－12　ライン終端の処理例



以上の手順を二次元符号化方式と呼びます。

１ページ分の符号化はKというパラメータを使って一次元符号化と二次元符号化の組み合わせで行います（**図１－13**）。

具体的な符号化は，次の手順で行います。

①　第１ライン目を一次元符号化します。
②　第２ライン目以降を二次元符号化します。
③　K＋１ライン目を一次元符号化します。
④　①から③の手順を繰り返します。

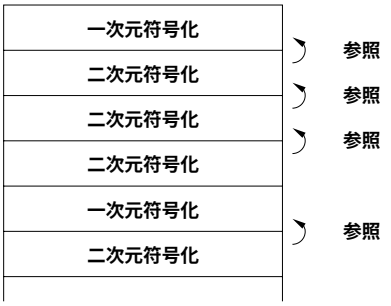
つまり，Kラインごとに一次元符号化方式で符号化します。これは次の理由によります。二次元符号化方式は直前のラインを参照して符号化します。このため，あるラインの符号が伝送誤りなどで正常な符号でなくなった場合，このライン以降のすべての復元画像が正常なものでなくなります。これを防ぐために，Kラインごとに一次元符号化を挿入します。

Kの値は，垂直方向の解像度により，一般的に次のようになります。

・100LPI（Lines Per Inch）の場合，K = 2
・200LPI（Lines Per Inch）の場合，K = 4

保守／廃止

図１－13　Kパラメータ（K＝4の場合）



一次元符号化したラインと二次元符号化したラインとを区別するため，ライン符号の区切り符号を付加します。

・一次元符号の直前：EOL＋1
・二次元符号の直前：EOL＋0

また，ページ終端には次の符号を付加します。

RTC=（EOL＋1）×6

この手順で符号化することにより，１ページ分の画像データは図１－14に示すようになります。

図１－14　１ページ分のMR符号

備考　RTC=（EOL＋1）×6

保守／廃止

1.2.4　MMR方式

MMR方式では，すべてのラインを二次元符号化します。ページの第１ライン目はその直前に仮想的な全白のラインがあるものとして，二次元符号化されます。

MR方式では，符号データを通常の電話回線で伝送することを前提として規定されているため，伝送誤りの伝播を制限するためにKパラメータが用いられています。これに対して，MMR方式はデジタル回線を用いた誤りのない伝送を前提としているため，Kパラメータを無限大に設定しています。

１ページ分の符号化は次の手順で行います。

① ページの第１ライン目はその直前に仮想的な全白のラインがあるものとして，第１ライン目からすべてのラインを二次元符号化します。各ライン符号間を区切る特別な符号は用いません。

② ページ終端を示す符号として，最終ライン符号の直後にEOFB（End Of Facsimile Block）符号を付加します。ただし，EOFB符号は次の値とします。

EOFB = EOL×2

③ １ページ分の符号データ量をバイト単位にそろえる必要がある場合，またはある大きさ以上にそろえる必要がある場合には，EOFB符号の後ろにパッド・ビットとして０のビット列を付加します。

この手順で符号化することにより，１ページ分の画像データは図１－15に示すようになります。

図１－15　１ページ分のMMR符号

備考　EOFB = EOL×2

保守／廃止

1.3　製品概要

1.3.1　特　　徴

① V55PI™のコーデック命令と同等のインタフェースを採用しています。

② 中断処理情報などの詳細な内部情報を公開しています。

中断処理情報：送信（受信）バッファに対して圧縮データを最後まで書き込んで（読み込んで）中断したときに，ライブラリから返される情報です。

③ NEC/GHS（Green Hills Software）製CコンパイラのC言語からの呼び出しができます。

④ NEC/GHS対応リアルタイムOSに対応（リエントラント可能）しています。

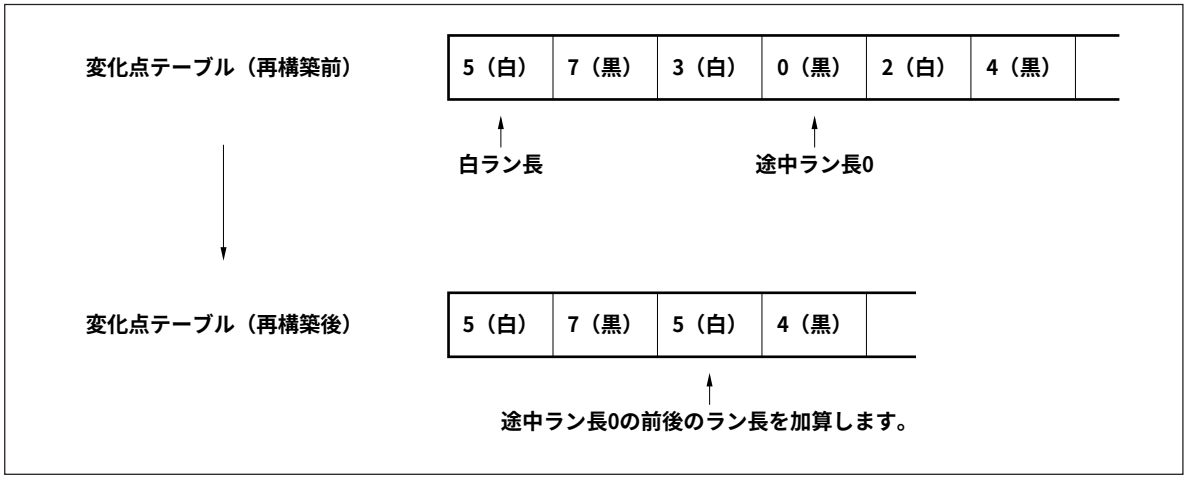
⑤ 復号化処理において，途中ラン長０検出時の変化点テーブル再構築処理をサポートしています。符号化処理は変化点テーブル再構築処理をサポートしていません。

途中ラン長０　：ライン先頭以外の復号時に連続する白または黒の画素数が０になることです。

変化点テーブル　：１ライン分のラン長情報です。

図１－16に途中ラン長０検出時の変化点テーブル再構築処理例を示します。

図１－16　途中ラン長０検出時の変化点テーブル再構築処理例



保守／廃止

1.3.2 機　能

★ （1）コア・ライブラリ

コア・ライブラリは，９つの関数を使用してMH/MR/MMRの機能を実現するライブラリです。各関数の組み合わせでMH/MR/MMR方式の圧縮／伸長処理を行います。

・圧縮処理系

指定された画像データ・バッファから１ライン分ごとに変化点テーブルを作成し，MH/MR/MMR方式で圧縮処理を行い，送出バッファへ符号データを出力します。

・伸長処理系

指定された受信バッファから，MH/MR/MMR方式で１ライン分の変化点テーブルを作成して伸長処理を行い，プリント・バッファへ画素データを出力します。

★ （2）APIライブラリ（APplication user Interface, AP30100-B01のみ）

APIライブラリは，ユーザ・アプリケーションとミドルウエアとのインタフェースを容易にするライブラリです。圧縮系／伸長系それぞれのコア・ライブラリを呼び出してMH/MR/MMR方式による圧縮／伸長処理を行います。

1.3.3 動作環境

★ ・動作対象CPU：VR4100シリーズ（AP30100-B01）注

V850ファミリ（AP703000-B01）

V830ファミリ（AP705100-B01 ）

V810ファミリ（AP70732-B01）

注 動作モード：32ビット・モード

エンディアン：リトル・エンディアン

保守／廃止

★　　・必要なメモリ・サイズ（AP30100-B01）

メモリの種類	内　　容		
ROM （約100 Kバイト）	プログラム	コア・ライブラリ	圧縮系：約6 Kバイト
			伸長系：約12 Kバイト
		APIライブラリ	圧縮系：約11 K（5 K＋6 K）バイト注1
			伸長系：約17 K（5 K＋12 K）バイト注1
	符号化／復号化テーブル	符号化テーブル	約21 Kバイト
		復号化テーブル	約33 Kバイト
RAM	バッファ	変化点テーブル注2	必要メモリ・サイズは，画像サイズ，送信／受信バッファ・サイズなどにより異なります。
		画素データ格納バッファ	
		送信／受信バッファ	
		プリント・バッファ	
	入出力パラメータ領域	コア・ライブラリ	圧縮系：58バイト
			伸長系：60バイト
		APIライブラリ注1	圧縮系：106（48＋58）バイト
			伸長系：108（48＋60）バイト
	スタック・エリア	コア・ライブラリ	圧縮系：約100バイト
			伸長系：約150バイト
		APIライブラリ注1	圧縮系：約250（150＋100）バイト
			伸長系：約300（150＋150）バイト

注１．APIライブラリのサイズ＝APIライブラリ本体のサイズ＋コア・ライブラリのサイズ

２．変化点テーブルの最低限必要サイズ：１ライン当たり＝（１ラインの画素数＋３）×２バイト

注意　メモリ・サイズは変更になる場合があります。

保守／廃止

・必要なメモリ・サイズ（AP703000-B01, AP705100-B01, AP70732-B01）

メモリの種類	内　　容		
ROM（約64 Kバイト）	プログラム（コア・ライブラリ）		圧縮系：約3 Kバイト
			伸長系：約4 Kバイト
	符号化／復号化テーブル	符号化テーブル	約22 Kバイト
		復号化テーブル	約35 Kバイト
RAM	バッファ	変化点テーブル注	必要メモリ・サイズは，画像サイズ，送信／受信バッファ・サイズなどにより異なります。
		画素データ格納バッファ	
		送信／受信バッファ	
		プリント・バッファ	
	入出力パラメータ＋内部情報領域		圧縮系：94バイト
			伸長系　92バイト（AP705100-B01, AP70732-B01）
			96バイト（AP703000-B01）

注　変化点テーブルの最低限必要サイズ：１ライン当たり＝（１ラインの画素数＋１）×２バイト

注意　メモリ・サイズは変更になる場合があります。

・サポート・ツール

★ VR4100シリーズ（AP30100-B01）
- GHS社製Ｃコンパイラ／アセンブラ
  - CC-MIPE（Windows版，SUN4™版）（Ver.1.8.8以上）

V850ファミリ（AP703000-B01）
- NEC製Ｃコンパイラ／アセンブラ
  - CA850（Windows版，SUN4版）（Ver.1.00以上）
- GHS社製Ｃコンパイラ／アセンブラ
  - CC850（Windows版，SUN4版）（Ver.1.8.7B以上）

V830ファミリ（AP705100-B01）
- NEC製Ｃコンパイラ／アセンブラ
  - CA830（Windows版，SUN4版）（Ver.1.00以上）
- GHS社製Ｃコンパイラ／アセンブラ
  - CC830（Windows版，SUN4版）（Ver.1.8.8以上）

V810ファミリ（AP70732-B01）
- NEC製Ｃコンパイラ／アセンブラ
  - CA732（Windows版，SUN4版）（Ver.2.00以上）
- GHS社製Ｃコンパイラ／アセンブラ
  - CC810（Windows版，SUN4版）（Ver.1.8.7B以上）

保守／廃止

1.3.4　パッケージ内容

パッケージ内容を次に示します。

★　（1）AP30100-B01（VR4100シリーズ）

```
/--- ghstools --+--〈lib4100〉---------- libmhmr.a    ：ライブラリ
                |
                +--〈smp4100〉---〈mr〉--- mh_enc.c     ：MH方式圧縮サンプル・メイン・ソース
                                          mr_enc.c     ：MR方式圧縮サンプル・メイン・ソース
                                          mmr_enc.c    ：MMR方式圧縮サンプル・メイン・ソース
                                          mh_dec.c     ：MH方式伸長サンプル・メイン・ソース
                                          mr_dec.c     ：MR方式伸長サンプル・メイン・ソース
                                          mmr_dec.c    ：MMR方式伸長サンプル・メイン・ソース
                                          mre_api.c    ：MR方式圧縮サンプル・メイン・ソース
                                                         （APIライブラリ使用）
                                          mrd_api.c    ：MR方式伸長サンプル・メイン・ソース
                                                         （APIライブラリ使用）
                                          crt4100.s    ：サンプル・スタートアップ・ルーチン
                                          codec_vr.h   ：ヘッダ・ファイル
                                          makenorm     ：メイク・ファイル（APIライブラリ未使用）
                                          makeapi      ：メイク・ファイル（APIライブラリ使用）
```

保守／廃止

（2）AP703000-B01（V850ファミリ）

NEC版とGHS版の２つがあり，オブジェクト・フォーマットが異なります。

```
root ┬ nectools ┬ 〈lib850〉 ---------- libmhmr.a     ：ライブラリ
     │           │
     │           └ 〈smp850〉 ― 〈mr〉 ― mh_enc.c     ：MH方式圧縮サンプル・メイン・ソース
     │                                   mr_enc.c     ：MR方式圧縮サンプル・メイン・ソース
     │                                   mmr_enc.c    ：MMR方式圧縮サンプル・メイン・ソース
     │                                   mh_dec.c     ：MH方式伸長サンプル・メイン・ソース
     │                                   mr_dec.c     ：MR方式伸長サンプル・メイン・ソース
     │                                   mmr_dec.c    ：MMR方式伸長サンプル・メイン・ソース
     │                                   crt850.s     ：サンプル・スタートアップ・ルーチン
     │                                   codec850.h   ：ヘッダ・ファイル
     │                                   makefile     ：メイク・ファイル
     │                                   mr_enc.lnk   ：リンク・ファイル
     │                                   mr_dec.lnk   ：リンク・ファイル
     │
     └ ghstools ┬ 〈lib850〉 ---------- libmhmr.a     ：ライブラリ
                │
                └ 〈smp850〉 ― 〈mr〉 ― mh_enc.c     ：MH方式圧縮サンプル・メイン・ソース
                                        mr_enc.c     ：MR方式圧縮サンプル・メイン・ソース
                                        mmr_enc.c    ：MMR方式圧縮サンプル・メイン・ソース
                                        mh_dec.c     ：MH方式伸長サンプル・メイン・ソース
                                        mr_dec.c     ：MR方式伸長サンプル・メイン・ソース
                                        mmr_dec.c    ：MMR方式伸長サンプル・メイン・ソース
                                        crt850.s     ：サンプル・スタートアップ・ルーチン
                                        reset.s      ：リセット・ベクタ
                                        codec850.h   ：ヘッダ・ファイル
                                        makefile     ：メイク・ファイル
```

（3）AP705100-B01（V830ファミリ）

NEC版とGHS版の２つがあり，オブジェクト・フォーマットが異なります。

```
root ┬ nectools ┬ 〈lib830〉 ---------- libmhmr.a     ：ライブラリ
     │           │
     │           └ 〈smp830〉 ― 〈mr〉 ― mh_enc.c     ：MH方式圧縮サンプル・メイン・ソース
     │                                  mr_enc.c     ：MR方式圧縮サンプル・メイン・ソース
     │                                  mmr_enc.c    ：MMR方式圧縮サンプル・メイン・ソース
     │                                  mh_dec.c     ：MH方式伸長サンプル・メイン・ソース
     │                                  mr_dec.c     ：MR方式伸長サンプル・メイン・ソース
     │                                  mmr_dec.c    ：MMR方式伸長サンプル・メイン・ソース
     │                                  crt830.s     ：サンプル・スタートアップ・ルーチン
     │                                  codec830.h   ：ヘッダ・ファイル
     │                                  makefile     ：メイク・ファイル
     │                                  mr_enc.lnk   ：リンク・ファイル
     │                                  mr_dec.lnk   ：リンク・ファイル
     │
     └ ghstools ┬ 〈lib830〉 ---------- libmhmr.a     ：ライブラリ
                │
                └ 〈smp830〉 ― 〈mr〉 ― mh_enc.c     ：MH方式圧縮サンプル・メイン・ソース
                                       mr_enc.c     ：MR方式圧縮サンプル・メイン・ソース
                                       mmr_enc.c    ：MMR方式圧縮サンプル・メイン・ソース
                                       mh_dec.c     ：MH方式伸長サンプル・メイン・ソース
                                       mr_dec.c     ：MR方式伸長サンプル・メイン・ソース
                                       mmr_dec.c    ：MMR方式伸長サンプル・メイン・ソース
                                       crt830.s     ：サンプル・スタートアップ・ルーチン
                                       reset.s      ：リセット・ベクタ
                                       codec830.h   ：ヘッダ・ファイル
                                       makefile     ：メイク・ファイル
```

保守／廃止

（4）AP70732-B01（V810ファミリ）

NEC版とGHS版の２つがあり，オブジェクト・フォーマットが異なります。

```
root ┬ nectools ┬〈lib732〉 ---------- libmhmr.a    ：ライブラリ
     │           │
     │           └〈smp732〉 ―〈mr〉 ―― mh_enc.c    ：MH方式圧縮サンプル・メイン・ソース
     │                                  mr_enc.c    ：MR方式圧縮サンプル・メイン・ソース
     │                                  mmr_enc.c   ：MMR方式圧縮サンプル・メイン・ソース
     │                                  mh_dec.c    ：MH方式伸長サンプル・メイン・ソース
     │                                  mr_dec.c    ：MR方式伸長サンプル・メイン・ソース
     │                                  mmr_dec.c   ：MMR方式伸長サンプル・メイン・ソース
     │                                  crt810.s    ：サンプル・スタートアップ・ルーチン
     │                                  codec810.h  ：ヘッダ・ファイル
     │                                  makefile    ：メイク・ファイル
     │                                  mr_enc.lnk  ：リンク・ファイル
     │                                  mr_dec.lnk  ：リンク・ファイル
     │
     └ ghstools ┬〈lib810〉 ---------- libmhmr.a    ：ライブラリ
                │
                └〈smp810〉 ―〈mr〉 ―― mh_enc.c    ：MH方式圧縮サンプル・メイン・ソース
                                       mr_enc.c    ：MR方式圧縮サンプル・メイン・ソース
                                       mmr_enc.c   ：MMR方式圧縮サンプル・メイン・ソース
                                       mh_dec.c    ：MH方式伸長サンプル・メイン・ソース
                                       mr_dec.c    ：MR方式伸長サンプル・メイン・ソース
                                       mmr_dec.c   ：MMR方式伸長サンプル・メイン・ソース
                                       crt810.s    ：サンプル・スタートアップ・ルーチン
                                       reset.s     ：リセット・ベクタ
                                       codec810.h  ：ヘッダ・ファイル
                                       makefile    ：メイク・ファイル
```

保守／廃止

1.3.5　目標性能

★　（1）AP30100-B01

条件：CPU　VR4100（40 MHz／外部10 MHz），32ビット・バス，
　　内蔵キャッシュ・メモリ使用（命令：2 Kバイト，データ：1 Kバイト）
　　Kファクタ = 4（MR方式）
　　データ　ITU-T chart1（1728×2376ドット）
性能：MH方式　圧縮時間　約0.18秒，伸長時間　約0.19秒
　　MR方式　圧縮時間　約0.23秒，伸長時間　約0.27秒
　　MMR方式　圧縮時間　約0.22秒，伸長時間　約0.25秒

（2）AP703000-B01

条件：CPU　V853（33 MHz），16ビット・バス
　　Kファクタ = 4（MR方式）
　　データ　ITU-T chart1（1728×2376ドット）
性能：MH方式　圧縮時間　約0.17秒，伸長時間　約0.15秒
　　MR方式　圧縮時間　約0.20秒，伸長時間　約0.20秒
　　MMR方式　圧縮時間　約0.20秒，伸長時間　約0.19秒

（3）AP705100-B01

条件：CPU　V830（100 MHz），32ビット・バス
　　Kファクタ = 4（MR方式）
　　データ　ITU-T chart1（1728×2376ドット）
性能：MH方式　圧縮時間　約0.06秒，伸長時間　約0.07秒
　　MR方式　圧縮時間　約0.07秒，伸長時間　約0.09秒
　　MMR方式　圧縮時間　約0.09秒，伸長時間　約0.08秒

（4）AP70732-B01

条件：CPU　V810（25 MHz），32ビット・バス，キャッシュ・オン
　　Kファクタ = 4（MR方式）
　　データ　ITU-T chart1（1728×2376ドット）
性能：MH方式　圧縮時間　約0.17秒，伸長時間　約0.20秒
　　MR方式　圧縮時間　約0.23秒，伸長時間　約0.28秒
　　MMR方式　圧縮時間　約0.24秒，伸長時間　約0.28秒

保守／廃止

第２章　ライブラリ仕様

２.１　処理概要

２.１.１　ライブラリ構成

表２－１に圧縮系ライブラリ，表２－２に伸長系ライブラリ，表２－３にAPIライブラリ[注]の一覧を示します。

注　AP30100-B01のみ

表２－１　圧縮系ライブラリー覧

関数名	機　　能	説　　明
mr_albit（）	データ送出処理	EOLコード，タグ・ビット，RTC，FILLビットなどのデータを送出します。
mr_coltrp（）	変化点テーブル作成処理	１ライン分の画素データの変化点テーブル作成を処理します。
mr_mhenc（）	MH符号化処理	１ライン分の変化点テーブルよりMH方式符号化を処理します。
mr_mrenc（）	MR符号化処理	参照ラインおよび符号化ラインの変化点テーブルよりMR方式符号化を処理します。

表２－２　伸長系ライブラリー覧

関数名	機　　能	説　　明
mr_scheol（）	EOL検出処理	符号化データからEOLコードを検出します。
mr_getbit（）	１ビット検出処理	符号化データからタグ・ビットを検出します。
mr_mhdec（）	MH復号化処理	符号データよりMH方式復号化を処理し，１ライン分の変化点テーブルを作成します。
mr_mrdec（）	MR復号化処理	符号データおよび参照ラインの変化点テーブルよりMR方式復号化を処理し，１ライン分の変化点テーブルを作成します。
mr_cnvtrp（）	画素データ作成処理	１ライン分の変化点テーブルより画素データを作成します。

★

表２－３　APIライブラリー覧

関数名	機　　能	説　　明
mr_encode（）	MH/MR/MMR符号化処理	画像データより指定したライン分の符号化を処理します。
mr_decode（）	MH/MR/MMR復号化処理	符号データより指定したライン分の復号化を処理します。

また，MH符号化方式，MR符号化方式における１ライン分の符号化／復号化処理の流れを図２－１，図２－２に，APIライブラリを使用した場合の符号化／復号化処理の流れを図２－３，図２－４に示します。

図２－１　１ライン分の符号化処理の流れ

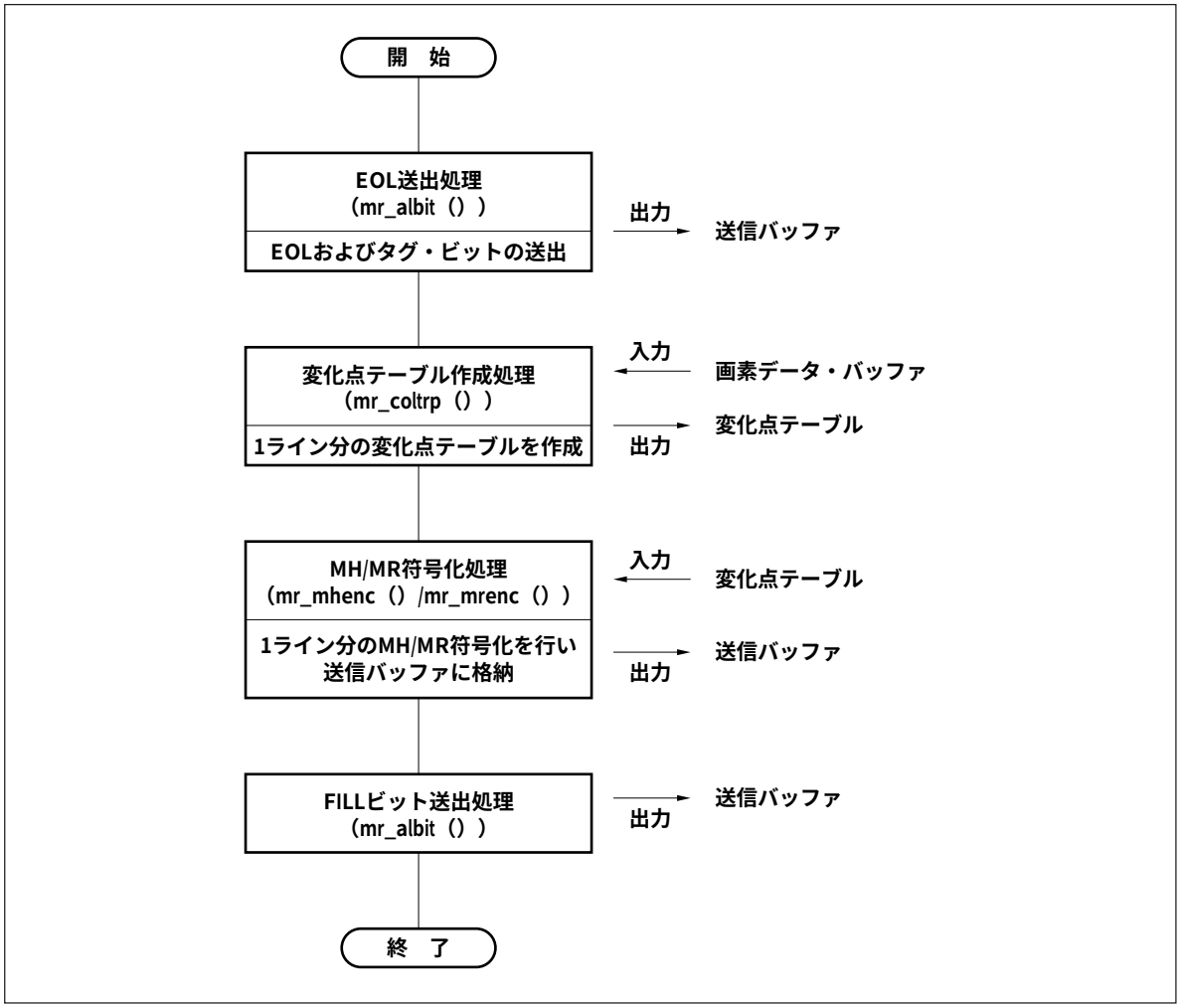
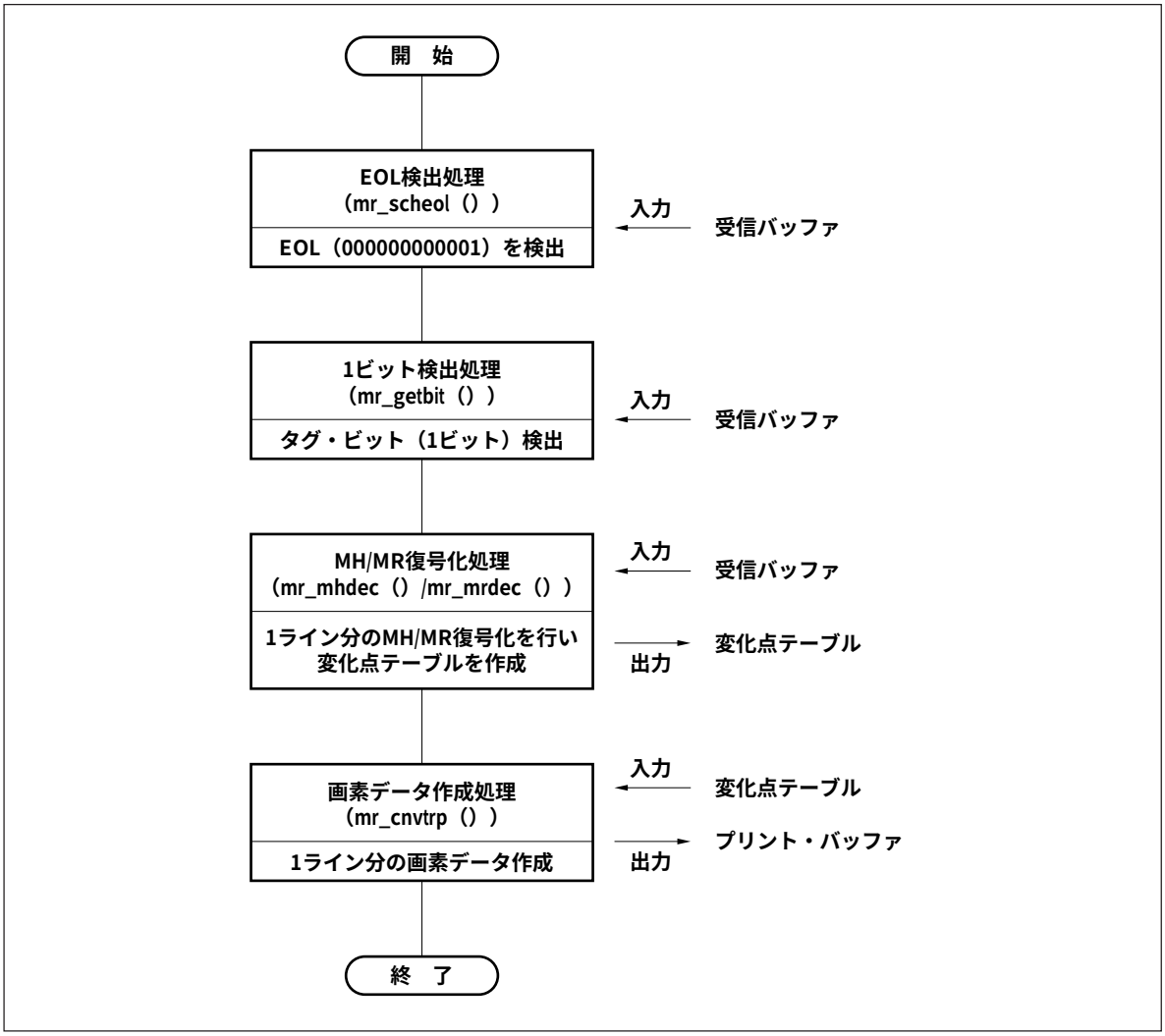
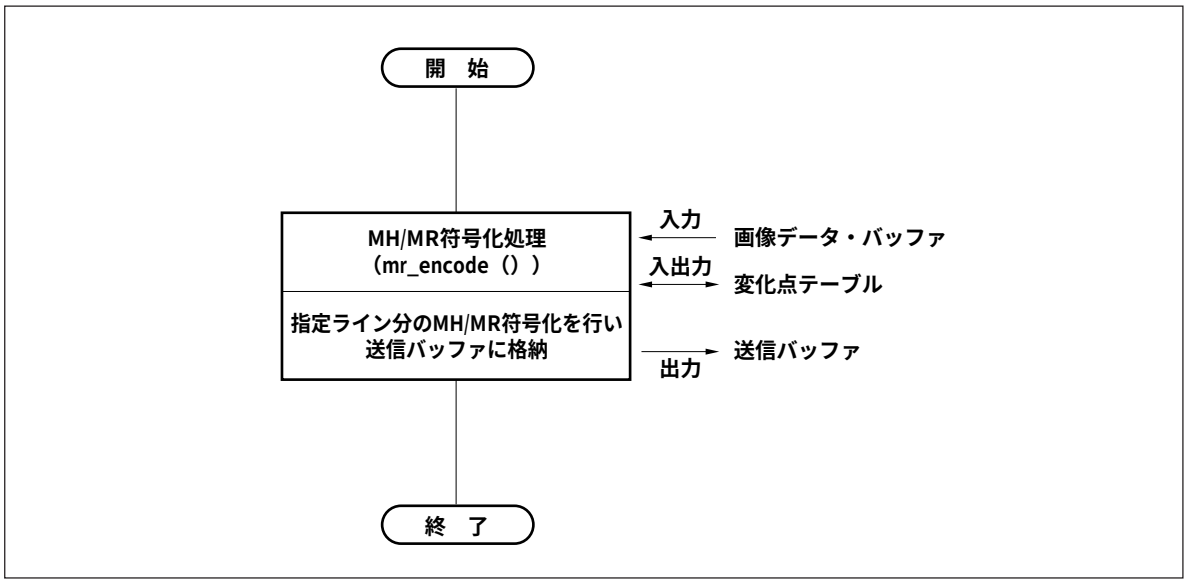


図２－２　１ライン分の復号化処理の流れ

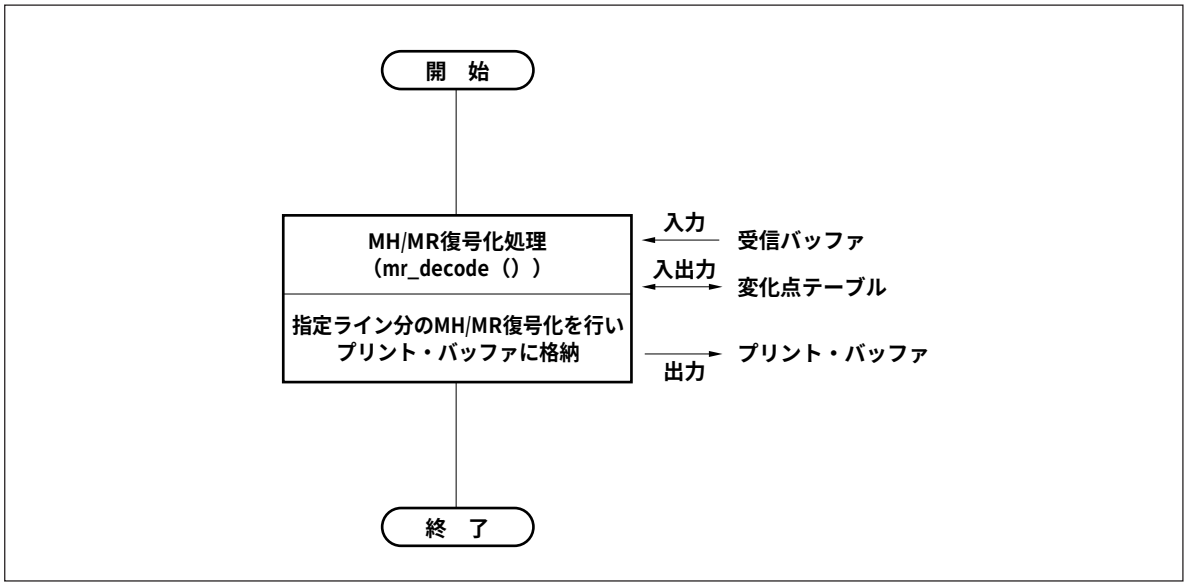


保守／廃止

★　図２－３　指定ライン分の符号化処理の流れ（APIライブラリを使用した場合）



★　図２－４　指定ライン分の復号化処理の流れ（APIライブラリを使用した場合）



保守／廃止

２.１.２　画素データと符号データの取り扱い

★ **（１）データの格納方式**

画像データ，符号データの格納方式を次に示します。

ビット順：LSB-first（バイトのLSBから順に格納）

バイト順：リトル・エンディアン（アドレスの小さい方から順に格納）

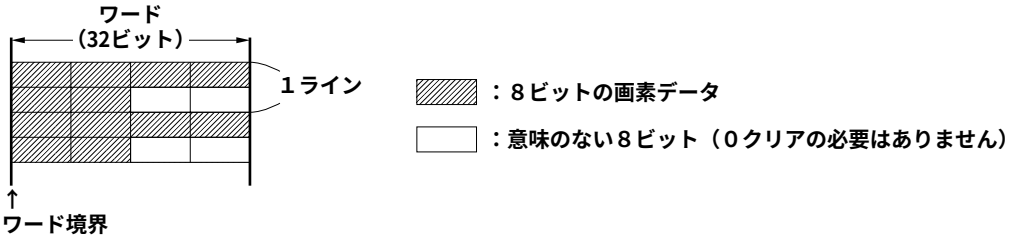
例　データの格納

アドレス	ビット							
	←MSB							LSB→
	7	6	5	4	3	2	1	0
0H	8	7	6	5	4	3	2	1 ←先頭データ
1H	16	15	14	13	12	11	10	9
2H	24	23	22	21	20	19	18	17
3H	32	31	30	29	28	27	26	25

（２）バッファ・アドレスとバッファの残りサイズの取り扱い

画素データ・バッファのアドレスには，ワードでアラインした値を入力してください。符号化対象画素データの１ラインの画素数が32の倍数でない場合，ライン先頭の画素データはワードでアラインされた位置に置いてください。

★ **例　１ライン48画素の場合**



（3）拡張モード

符号化処理では，拡張メークアップ・コードとダブル・メークアップをサポートします（最大2560×2＋63＝5183ビット）。復号化処理では制限はありません。

（4）非圧縮モード

非圧縮モードはサポートしていません。エラーとして検出します。

（5）32ビットに満たない符号データの取り扱い

32ビットに満たない符号データは圧縮系と伸長系で取り扱いが異なります。圧縮系と伸長系のそれぞれについて次に説明します。

（a）圧 縮 系

このライブラリでは，32ビットに満たない符号データ，または符号データのうち32ビットを越えた部分は，処理中半端符号データ（odd_data）のLSBに格納され保持されます。次に送出したい符号データとあわせて32ビット以上になったところで指定された送信バッファへ出力します。そのビット数は処理中半端符号データ・ビット数（odd_bit）に保持します。

次に，32ビットに満たない場合と32ビット以上の場合の，処理中半端符号データ（odd_data），処理中半端符号データ・ビット数（odd_bit），送信バッファの関係を示します。

保守／廃止

図２－５　処理中半端符号と送出符号をあわせて32ビットに満たない場合

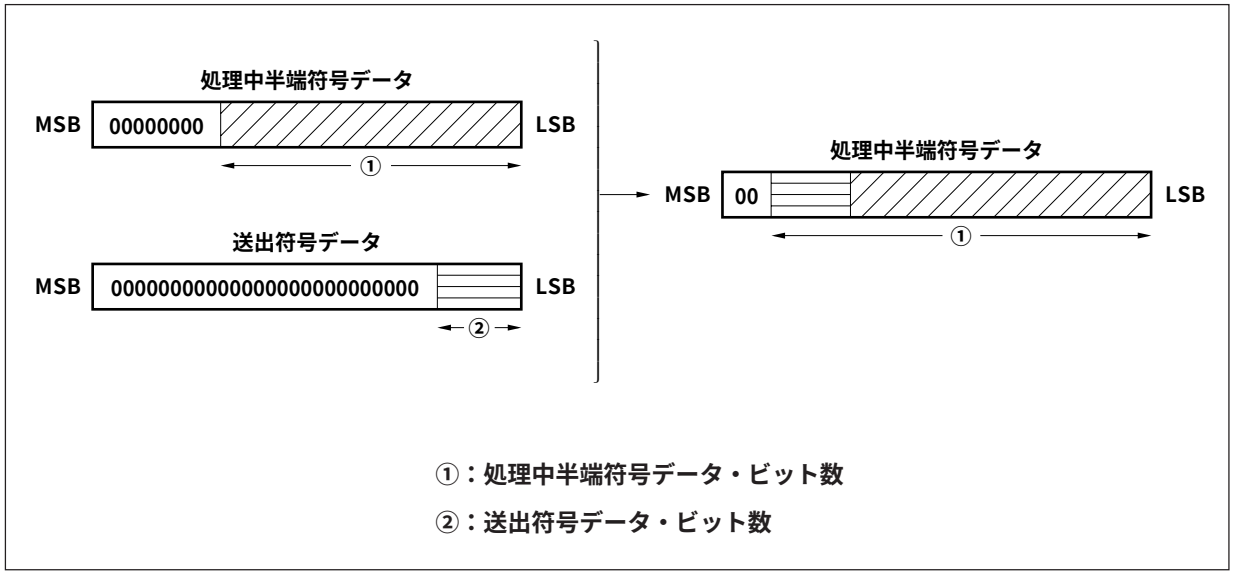
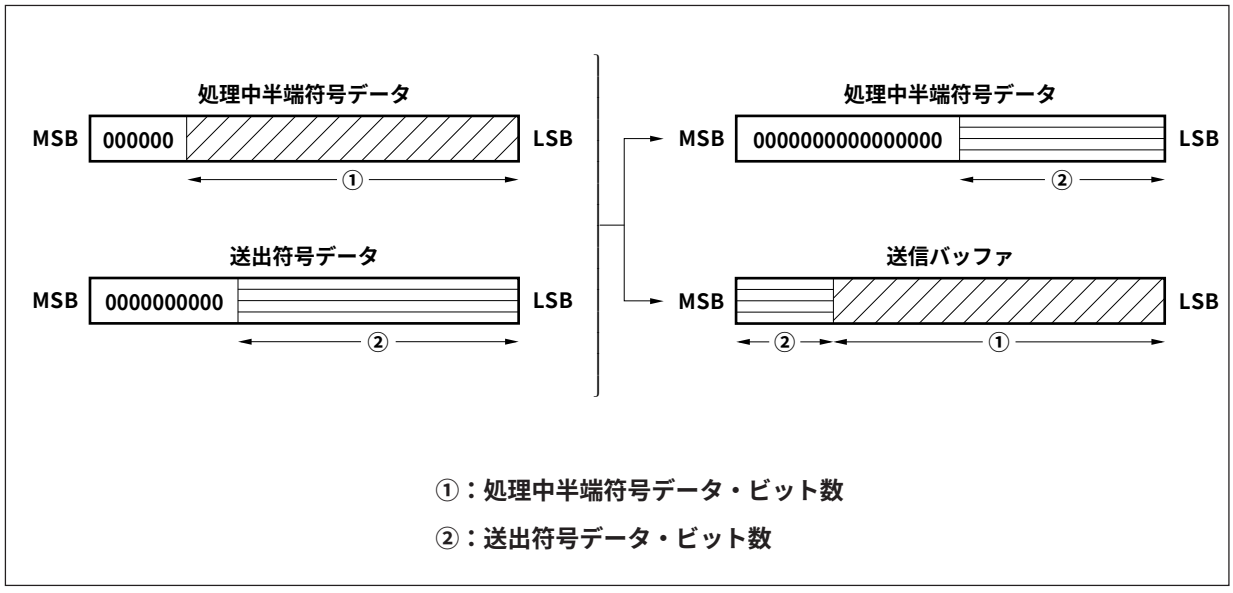


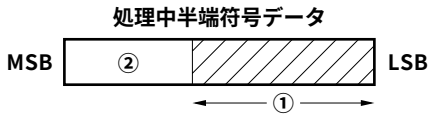
図２－６　処理中半端符号と送出符号をあわせて32ビット以上の場合



（b）伸 長 系

このライブラリでは，伸長対象となる32ビットの符号データは，受信バッファから処理中半端符号データ（odd_data）のLSBに格納され保持されます。復号し終えたデータ長は処理中半端符号データ・ビット数（odd_bit）として保持されます。

図２－７　処理中半端符号データ／処理中半端符号データ・ビット数の関係



①：処理中半端符号データ・ビット数（復号が終わったデータのビット数）

②：まだ復号が終わっていないデータ

保守／廃止

★　２.２　関数仕様（AP30100-B01）

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。
入出力パラメータ＋内部情報領域は，圧縮／伸長系で次の構造体で定義されます。

２.２.１　AP30100-B01の構造体（パラメータ）

MH/MR/MMRの圧縮／伸長の関数で使用しているパラメータ（MREINFO, MRDINFO, MRAPIINFO）を次に説明します。

（１）圧 縮 系

領域：58バイト

パラメータ：MREINFO

	メンバ	型	説　　明
入出力パラメータ	odd_data	unsigned int	処理中半端符号データ
	send_data	unsigned int	送出符号データ
	odd_bit	unsigned char	処理中半端符号データ・ビット数
	send_bit	unsigned char	送出符号データ・ビット数
	pixel_num	unsigned short	１ラインの画素数
	enc_buf_size	unsigned int	送信バッファ残りサイズ
	*enc_buf	unsigned int	送信バッファ・アドレス
	*pixel_buf	unsigned int	画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	再開アドレス／初期化フラグ
	bit_length	unigned short	ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unigned short	a0ポジション（〃）
	a1pos	unigned short	a1ポジション（〃）
	b0pos	unigned short	b0ポジション（〃）
	b1pos	unigned short	b1ポジション（〃）
	b2pos	unigned short	b2ポジション（〃）
	exp_length	unigned short	拡張コード時残りレングス（〃）
	a0_color	unigned char	色（白黒）情報（〃）
	bit_offset	unigned char	ビット・サーチ・ワード内オフセット（〃）

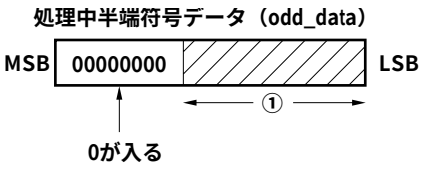
各メンバについて，次に説明します。

・odd_data

注意　圧縮処理／伸長処理で取り扱いが異なります。

符号化の終了した符号データで32ビットに満たない部分を処理中半端符号データとしてLSBから格納します。詳細は**２．１．２（５）32ビットに満たない符号データの取り扱い**を参照してください。
符号データのデータ長は処理中半端符号データ・ビット数（odd_bit）となります。

図２－８　圧縮系の処理中半端符号データとデータ・ビット数の関係（AP30100-B01）



①：処理中半端符号データ・ビット数（符号データ長）

・send_data

EOLやFILLなど，ユーザが符号データを送出するときに設定します。
符号データ長は32ビット以下です。
処理中半端符号データ（odd_data）と合わせます。
詳細は**２．１．２（５）32ビットに満たない符号データの取り扱い**を参照してください。

・odd_bit

処理中半端符号データ（odd_data）中の処理中半端符号データ・ビット数（符号化の終了した符号データ・ビット数）を示します。

・send_bit

送出符号データ（send_data）の送出符号データ・ビット数を示します。

・pixel_num

１ラインの画素数を示します。最大5183まで指定できます。

・enc_buf_size

送信バッファの残りサイズを示します。

処理中半端符号データが32ビット以上になり，送信バッファへ書き込まれるとデクリメント（－４）されます。このメンバが０になると処理を中断します。
このメンバには，ワードでアラインした値を入力してください。

- *enc_buf

送信バッファのアドレスを示します。
処理中半端符号データが32ビット以上になり，送信バッファへ書き込まれるとインクリメント（＋４）されます。
このメンバには，ワードでアラインした値を入力してください。

- *pixel_buf

画像データのアドレスを示します。
画像データが32画素分圧縮されるとインクリメント（＋４）されます。
このメンバには，ワードでアラインした値を入力してください。

- *ref_tbl

参照ライン用の変化点テーブルのアドレスを示します。
このメンバには，ハーフワードでアラインした値を入力してください。

- *run_tbl

走査ライン用の変化点テーブルのアドレスを示します。
画素データがラン長に変換された場合などにインクリメント（＋２）されます。
このメンバには，ハーフワードでアラインした値を入力してください。

- run_tbl_buf_size

変化点テーブル・バッファ残りサイズを示します。
mr_coltrp（）においてデクリメント（－２）されます。
サイズはハーフワードでアラインした値で指定してください。０が入力された場合は10000Hが設定されます。
１ライン処理中にこのパラメータが０になると，処理を中断します。

- restart_adr

各ライブラリの処理中に中断した場合の再開アドレスを示します。
正常終了のときは０を返します。
中断情報を無視し，ライブラリの処理を初期化したいときは０を設定してください。

・**bit_length**

mr_coltrp（）において中断が発生したとき，１ラインの残り画素データをビット・サーチする残りレングスを示します。

・**a0pos**

mr_mhenc（），mr_mrenc（）において中断が発生したときのa0ポジションを示します。

・**a1pos**

mr_mrenc（）において中断が発生したときのa1ポジションを示します。

・**b0pos**

mr_mrenc（）において中断が発生したときのb0ポジションを示します。

・**b1pos**

mr_mrenc（）において中断が発生したときのb1ポジションを示します。

・**b2pos**

mr_mrenc（）において中断が発生したときのb2ポジションを示します。

・**exp_length**

mr_mhenc（），mr_mrenc（）において中断が発生したときの拡張コード時残りレングスを示します。

・**a0_color**

mr_mrenc（）において中断が発生したときのa0ポジションの色（白黒）情報を示します。
（0H：白，FFH：黒）

・**bit_offset**

mr_coltrp（）において中断が発生したとき，画素データをビット・サーチしたワード内のオフセット値を示します。

（2）伸 長 系

領域：60バイト

パラメータ：MRDINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	処理中半端符号データ
	pixel_data	unsigned int	処理中半端画素データ
	odd_bit	unsigned char	処理中半端符号データ・ビット数
	pixel_bit	unsigned char	処理中半端画素データ・ビット数
	pixel_num	unsigned short	１ラインの画素数
	dec_buf_size	unsigned int	受信バッファ残りサイズ
	*dec_buf	unsigned int	受信バッファ・アドレス
	*print_buf	unsigned int	プリント・バッファ・アドレス
	*ref_tbl	unsigned short	参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	再開アドレス／初期化フラグ
	zero_cnt	unsigned short	０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	b1ポジション（〃）
	run_M_len	unsigned short	メークアップ・ラン長（〃）
	run_pass_len	unsigned short	パス・モード時のラン長（〃）
	chg_cnt	unsigned short	水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	ラン長０検出フラグ（〃）
	a0_color	unsigned char	色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	中断ステータス（０：受信バッファ，１：変化点テーブル）

各メンバについて次に説明します。

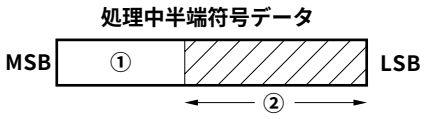
・odd_data

注意　圧縮処理／伸長処理で意味が異なります。

伸長対象となる32ビットの符号データを，受信バッファから処理中半端符号データ（odd_data）にLSBから格納し保持します。伸長処理が終了した符号データ長は処理中半端符号データ・ビット数（odd_bit）に保持します。

保守／廃止

図２－９　伸長系の処理中半端符号データとデータ・ビット数の関係（AP30100-B01）



①：まだ復号していないデータ

②：処理中半端符号データ・ビット数（復号が終わったデータ長）

・pixel_data

伸長処理後の32ビットに満たない画素データを保持します。

・odd_bit

処理中半端符号データ（odd_data）中の処理中半端符号データ・ビット数（伸長処理が終了した符号データ・ビット数）を示します。

・pixel_bit

処理中半端画素データ（pixel_data）中の画素データ・ビット数を示します。

・pixel_num

１ラインの画素数を示します。

・dec_buf_size

受信バッファの残りサイズを示します。

受信バッファから符号データを読み出すとデクリメント（－4）されます。このメンバが０のとき，またはこのメンバが４，さらに処理中符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。

このメンバには，ワードでアラインした値を入力してください。

・*dec_buf

受信バッファのアドレスを示します。

処理中半端符号データへ送出するとインクリメント（＋4）されます。

このメンバには，ワードでアラインした値を入力してください。

・*print_buf

画像データのアドレスを示します。

画像データが32画素分伸長されるとインクリメント（＋4）されます。

このメンバには，ワードでアラインした値を入力してください。

・*ref_tbl

参照ライン用の変化点テーブルのアドレスを示します。

このメンバには，ハーフワードでアラインした値を入力してください。

・*run_tbl

走査ライン用の変化点テーブルのアドレスを示します。

符号データがラン長に変換された場合などにインクリメント（＋2）されます。

このメンバには，ハーフワードでアラインした値を入力してください。

・run_tbl_buf_size

変化点テーブル・バッファ残りサイズを示します。

mr_mhdec（），mr_mrdec（）においてデクリメント（－2）されます。

サイズはハーフワードでアラインした値で指定してください。０が入力された場合は10000Hが設定されます。

１ライン処理中にこのパラメータが０になると，処理を中断します。

・restart_adr

各ライブラリの処理中に中断した場合の再開アドレスを示します。

初期化のときは，０に設定してください。

・zero_cnt

mr_scheol（）において中断が発生したときのEOLコードの０の数を保持します。

内部では32ビット・レジスタを使用していますが，入出力のパラメータとしては上位16ビットをマスクして出力しています。ただし，動作上は保証しています。

・a0pos

mr_mhdec（），mr_mrdec（）において，中断発生時および異常コード検出時のa0ポジションを示します。

・b1pos

mr_mrdec（）において，中断発生時および異常コード検出時のb1ポジションを示します。

・run_M_len

mr_mhdec（），mr_mrdec（）において，メークアップ・コード復号後の中断発生時および異常コード検出時のメークアップ・コードのラン長を示します。

保守／廃止

・**run_pass_len**

mr_mrdec（）において，パス・モード復号後の中断発生時および異常コード検出時のパス・モードのラン長を示します。

・**chg_cut**

mr_mrdec（）において，水平モード復号中の中断発生時および異常コード検出時の水平モードの変化点テーブル要素数を示します。

・**run_tbl_sadr**

mr_mhdec（），mr_mrdec（）において，中断が発生したときの走査ライン用変化点テーブルの先頭アドレスを示します。

・**run0_flag**

mr_mhdec（），mr_mrdec（）において，中断が発生したときの変化点テーブル再構築用のラン長０検出フラグを示します。（０：ラン長０データなし，１：ラン長０データあり）

・**a0_color**

mr_mrdec（）において，中断発生時および異常コード検出時のa0ポジションの色（白，黒）情報を示します。（0H：白，FFH：黒）

・**int_sts**

mr_mhdec（），mr_mrdec（）において，中断が発生した場合の中断の原因を示します。

０：受信バッファ残りサイズがない（dec_buf_size = 0 or（dec_buf_size = 4 and odd_bit＜13））

１：変化点テーブル・バッファ残りサイズがない（run_tbl_buf_size = 0）

保守／廃止

（３）MRAPIINFO

MRAPIINFO構造体は，APIライブラリの入出力パラメータが格納された構造体で，APIライブラリの圧縮／伸長の関数で共通して使用します。

メンバ		型	説　　明
入出力パラメータ	page_line	unsigned int	１ページのライン数
	pixel_line	unsigned int	処理ライン数
	line_cnt	unsigned int	累積ライン数カウンタ
	*run_len_tbl	unsigned short	変化点テーブル先頭アドレス
	reset	unsigned char	１：リセット実行
	method	unsigned char	方式（１：MH，２：MR，３：MMR）
	k_para	unsigned char	Ｋパラメータ
	rtc_cnt	unsigned char	RTCのEOLカウンタ
	fill_bit	unsigned int	最小送出ビット数
	restart_adr	unsigned int	再開アドレス
	mr_val［5］	unsigned int	変数退避エリア（中断時専用）

各メンバについて次に説明します。

・**page_line**

１ページのライン数を示します。

圧縮時のみ使用します。ページ先頭で１ページのライン数が不明のときは，FFFF FFFFHを設定してください。ライン数が確定したときに，再設定可能です。ただし，累積ライン数カウンタ（line_cnt）より小さい値を設定した場合の動作は保証できません。

・**pixel_line**

処理ライン数を示します。

指定ライン分の処理を終了した場合は０を出力します。

中断時は，残りのライン数を出力します。

・**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。

ライン数が不明なときに参照してください。reset = 1によってクリアされます。

• ***run_len_tbl**

変化点テーブル先頭アドレスを示します。

このアドレスから変化点テーブルの最低限必要サイズ分の領域を変化点テーブルとして使用します。使用するサイズは方式により異なります。次に使用するサイズを示します。

方式	変化点テーブルの使用サイズ
MH	１ライン分使用
MR	２ライン分使用
MMR	２ライン分使用

備考　１ライン当たりの使用サイズ＝（１ラインの画素数＋３）×２バイト

• **reset**

ライブラリを初期化します。

リセット（reset = 1）でライブラリを実行すると，次のメンバとメモリ領域を初期化します。

line_cnt, restart_adr:0

参照ライン用変化点テーブル（MMR時のみ）：オール白の変化点情報

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。

• **method**

１ページの符号化／復号化方式を示します。

methodの値と方式の対応は次に示すとおりで，ページ処理中に変更することはできません。

methodの値	方　　式
1	MH
2	MR
3	MMR

• **k_para**

MR方式の圧縮時におけるＫパラメータの値を示します。

１から255の値を設定できます。一般的にＫパラメータの値には４を設定します。

0が設定された場合の動作は，保証できません。

• **rtc_cnt**

RTCとして付加／検出するEOL/（EOL＋1）のカウント値を示します。

・圧縮処理時

ページの先頭でrtc_cntに次の値を設定してください。

方　　式	RTCの値	rtc_cntへの設定値
MH	EOL×6	6
MR	（EOL＋1）×6	6
MMR	EOL×2	2

備考　MMR方式では，EOFB（End Of Facsimile Block）符号を付加します。

ページ終了時にRTCを付加します。０を設定した場合には，RTCを付加しません。

RTCの付加中に中断が起こった場合には，残りのカウント値を出力します。RTCの付加が終了すると０を出力します。

・伸長処理時

RTCの検出中に中断が起こった場合には，残りのカウント値を出力します。RTCの検出が終了すると０を出力します。

・fill_bit

１ライン当たりの最小送出ビット数を示します。圧縮時のみ使用します。

１ラインのEOL＋符号のビット数が，最小送出ビット数（fill_bit）より少ないときにフィル・ビットを付加します。MMR方式の場合は無効になります。

・restart_adr

各ライブラリの処理中に中断した場合の再開アドレスを示します。

restart_adr = 0では，指定ライン数の先頭として処理を行います。reset = 1によりクリアされます。

・mr_val［5］

中断時に変数退避エリアとして使用します。

保守／廃止

２.２.２　外部インタフェース（AP30100-B01）

（1）データ送出

・分　　類　　圧縮処理系

・関 数 名　　mr_albit（）

・機能概要　　EOL, FILLなど32ビット以下のデータを送信バッファへ出力します。

・形　　式　　#include ” codec_vr.h ”
int mr_albit（MREINFO *mreinfo）

・引 き 数　　MREINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	send_data	unsigned int	(i)　送出符号データ
	odd_bit	unsigned char	(io) 処理中半端符号データ　・ビット数
	send_bit	unsigned char	(i)　送出符号データ・ビット数
	pixel_num	unsigned short	(-)　１ラインの画素数
	enc_buf_size	unsigned int	(io) 送信バッファ残りサイズ
	*enc_buf	unsigned int	(io) 送信バッファ・アドレス
	*pixel_buf	unsigned int	(-)　画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(-)　走査ライン用変化点テーブル・アドレス
	ref_tbl_buf size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(-)　再開アドレス／初期化フラグ
	bit_length	unigned short	(-)　ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unigned short	(-)　a0ポジション（〃）
	a1pos	unigned short	(-)　a1ポジション（〃）
	b0pos	unigned short	(-)　b0ポジション（〃）
	b1pos	unigned short	(-)　b1ポジション（〃）
	b2pos	unigned short	(-)　b2ポジション（〃）
	exp_length	unigned short	(-)　拡張コード時残りレングス（〃）
	a0_color	unigned char	(-)　色（白黒）情報（〃）
	bit_offset	unigned char	(-)　ビット・サーチ・ワード内オフセット（〃）

備考　ｉ：入力，ｏ：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断

・機　　能　　EOL, FILLなど32ビット以下のデータを送信バッファへ出力する処理を行います。送信バッファ残りサイズが０になると中断します。
処理中半端符号データについては**２．１．２（５）32ビットに満たない符号データの取り扱い**を参照してください。

注意　送信バッファの残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください。（入力できる値については２．２．１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（２）変化点テーブル作成

・分　　類　　圧縮処理系

・関 数 名　　mr_coltrp（）

・機能概要　　１ライン分の画素データの変化点テーブルを指定されたメモリ上に生成します。

・形　　式　　#include ” codec_vr.h ”
int mr_coltrp（MREINFO *mreinfo）

・引 き 数　　MREINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(-)　処理中半端符号データ
	send_data	unsigned int	(-)　送出符号データ
	odd_bit	unsigned char	(-)　処理中半端符号データ　・ビット数
	send_bit	unsigned char	(-)　送出符号データ・ビット数
	pixel_num	unsigned short	(i)　１ラインの画素数
	enc_buf_size	unsigned int	(-)　送信バッファ残りサイズ
	*enc_buf	unsigned int	(-)　送信バッファ・アドレス
	*pixel_buf	unsigned int	(io) 画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(io) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	bit_length	unigned short	(io) ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unigned short	(-)　a0ポジション（〃）
	a1pos	unigned short	(-)　a1ポジション（〃）
	b0pos	unigned short	(-)　b0ポジション（〃）
	b1pos	unigned short	(-)　b1ポジション（〃）
	b2pos	unigned short	(-)　b2ポジション（〃）
	exp_length	unigned short	(-)　拡張コード時残りレングス（〃）
	a0_color	unigned char	(-)　色（白黒）情報（〃）
	bit_offset	unigned char	(io) ビット・サーチ・ワード内オフセット（〃）

備考　ｉ：入力，ｏ：出力，io：入出力，-：未使用

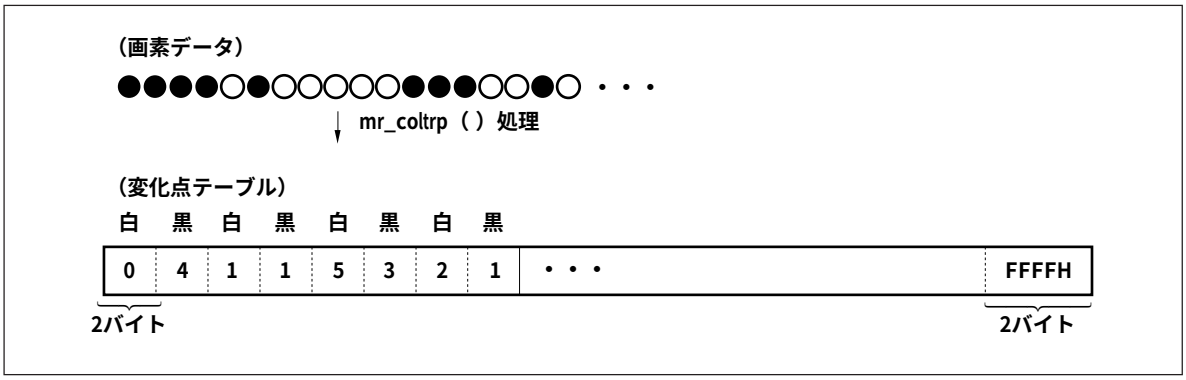
保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断（変化点テーブル・サイズ・オーバフロー）

・機　　能　　符号化処理において，１ライン分の画素データの変化点テーブルを指定されたメモリ上に生成します。１ライン処理中に変化点テーブル残りサイズが０になると中断します。変化点テーブル・サイズの指定はハーフワードでアラインした値が必要になります。変化点テーブル・サイズに０が入力された場合は10000Hが設定されます。
変化点テーブルは白のラン・レングスを先頭に２バイトごとに各色（白／黒）のラン・レングスを順に格納します。変化点テーブルの最終２バイトには１ライン分の終了情報として“ FFFFH ”を格納します。初期化のときに再開アドレス／初期化フラグを０にしてください。中断後の再開処理はできません。ラインの先頭からの再設定が必要です。
次に，変化点テーブルのフォーマットを示します。

図２－10　変化点テーブルのフォーマット（AP30100－B01）



１ライン処理で中断させないためには，変化点テーブル領域として（１ライン画素数＋３）×２バイトのメモリを確保してください。

保守／廃止

（３）MH符号化

・分　　類　　圧縮処理系

・関 数 名　　mr_mhenc（）

・機能概要　　１ライン（走査ライン）分の画素データの変化点情報（変化テーブル）を入力し，MH符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。

・形　　式　　#include ” codec_vr.h ”
int mr_mhenc（MREINFO *mreinfo）

・引 き 数　　MREINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	send_data	unsigned int	(-)　送出符号データ
	odd_bit	unsigned char	(io) 処理中半端符号データ　・ビット数
	send_bit	unsigned char	(-)　送出符号データ・ビット数
	pixel_num	unsigned short	(i)　１ラインの画素数
	enc_buf_size	unsigned int	(io) 送信バッファ残りサイズ
	*enc_buf	unsigned int	(io) 送信バッファ・アドレス
	*pixel_buf	unsigned int	(-)　画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	bit_length	unigned short	(-)　ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unigned short	(io) a0ポジション（〃）
	a1pos	unigned short	(-)　a1ポジション（〃）
	b0pos	unigned short	(-)　b0ポジション（〃）
	b1pos	unigned short	(-)　b1ポジション（〃）
	b2pos	unigned short	(-)　b2ポジション（〃）
	exp_length	unigned short	(io) 拡張コード時残りレングス（〃）
	a0_color	unigned char	(-)　色（白黒）情報（〃）
	bit_offset	unigned char	(-)　ビット・サーチ・ワード内オフセット（〃）

備考　ｉ：入力，ｏ：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に符号化する変化点情報が存在

・機　　能　　１ライン（走査ライン）分の画素データの変化点情報（変化点テーブル）を入力し，MH符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。１ライン符号化の終了条件は，変化点テーブルのターミネータ（FFFFH）の検出時です。１ライン処理中に送信バッファ残りサイズが０になると中断します。初期化のとき，再開アドレス／初期化フラグを０に設定してください。

注意　送信バッファの残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください（入力できる値については２.２.１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（４）MR符号化

・分　　類　　圧縮処理系

・関 数 名　　mr_mrenc（）

・機能概要　　参照ライン／走査ラインの変化点情報（変化点テーブル）を入力し，１ライン分のMR符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。

・形　　式　　#include ” codec_vr.h ”
int mr_mrenc（MREINFO *mreinfo）

・引 き 数　　MREINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	send_data	unsigned int	(-)　送出符号データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	send_bit	unsigned char	(-)　送出符号データ・ビット数
	pixel_num	unsigned short	(i)　１ラインの画素数
	enc_buf_size	unsigned int	(io) 送信バッファ残りサイズ
	*enc_buf	unsigned int	(io) 送信バッファ・アドレス
	*pixel_buf	unsigned int	(-)　画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(io) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	bit_length	unsigned short	(-)　ビット・サーチ残りレングス（中断処理時使用）
	a0pos	unsigned short	(io) a0ポジション（〃）
	a1pos	unsigned short	(io) a1ポジション（〃）
	b0pos	unsigned short	(io) b0ポジション（〃）
	b1pos	unsigned short	(io) b1ポジション（〃）
	b2pos	unsigned short	(io) b2ポジション（〃）
	exp_length	unsigned short	(io) 拡張コード時残りレングス（〃）
	a0_color	unsigned char	(io) 色（白黒）情報（〃）
	bit_offset	unsigned char	(-)　ビット・サーチ・ワード内オフセット（〃）

備考　ｉ：入力，ｏ：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に符号化する変化点情報が存在

・機　　能　　参照ライン／走査ラインの変化点情報（変化点テーブル）を入力し，１ライン分のMR符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。１ライン符号化の終了条件は，走査用変化点テーブルのターミネータ（FFFFH）の検出時です。

１ライン処理中に送信バッファ残りサイズが０になると中断します。初期化のときに再開アドレス／初期化フラグを０に設定してください。

注意　送信バッファの残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください（入力できる値については２.２.１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（５）EOL検出

・分　　類　　伸長処理系

・関 数 名　　mr_scheol（）

・機能概要　　メモリ上（受信バッファ）の指定された位置からEOLを検出します。

・形　　式　　#include ” codec_vr.h ”
int mr_scheol（MRDINFO *mrdinfo）

・引 き 数　　MRDINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixel_data	unsigned int	(-) 処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixel_bit	unsigned char	(-) 処理中半端画素データ・ビット数
	pixel_num	unsigned short	(-) １ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*print_buf	unsigned int	(-) プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(-) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	zero_cnt	unsigned short	(io) ０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(-) a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-) b1ポジション（〃）
	run_M_len	unsigned short	(-) メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-) パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-) 水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(-) 走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(-) ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-) 色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(-) 中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　ｉ：入力，ｏ：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
01	中断
04	先頭にEOLを検出（正常終了）
05	FILL付きEOLを先頭に検出（正常終了）
12	EOL以外のコードを検出した後，EOLを検出

・機　　能　　復号化処理において，メモリ上（受信バッファ）の指定された位置からEOLを検出します。指定された受信バッファ内にEOLが検出されない場合に中断します。
初期化のとき，再開アドレス／初期化フラグを０にしてください。
このライブラリでは，受信バッファ残りサイズが０のとき，または受信バッファ残りサイズが４，さらに処理中半端符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。

注意１．受信バッファの残りサイズが4Hで，処理中半端符号データ・ビット数が13ビット未満にEOLコードが存在しても中断します。
２．０ビット・カウンタは，内部では32ビット・レジスタを使用していますが，入出力パラメータの出力としては上位16ビットをマスクして出力しています。ただし，動作上は保証しています。

（6）１ビット検出

・分　　類　　伸長処理系

・関 数 名　　mr_getbit（）

・機能概要　　タグ・ビットなどの１ビットを検出します。

・形　　式　　＃include ” codec_vr.h ”
int mr_getbit（MRDINFO *mrdinfo）

・引 き 数　　MRDINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixel_data	unsigned int	(-) 処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixel_bit	unsigned char	(-) 処理中半端画素データ・ビット数
	pixel_num	unsigned short	(-) １ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*print_buf	unsigned int	(-) プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(-) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(-) 再開アドレス／初期化フラグ
	zero_cnt	unsigned short	(-) ０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(-) a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-) b1ポジション（〃）
	run_M_len	unsigned short	(-) メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-) パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-) 水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(-) 走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(-) ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-) 色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(-) 中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　ｉ：入力，ｏ：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	タグ・ビットが０
01	タグ・ビットが１

・機　　能　　復号化処理において，タグ・ビットなどの１ビットを検出します。メモリ上（受信バッファ）の指定された位置から１ビットを取り出し，関数の返り値に設定します。

注意　送信バッファの残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください（入力できる値については２.２.１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（７）MH復号化

・分　　類　　伸長処理系

・関 数 名　　mr_mhdec（）

・機能概要　　指定されたメモリ上（受信バッファ）の符号データを入力し，MH復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

・形　　式　　#include ” codec_vr.h ”
int mr_mhdec（MRDINFO *mrdinfo）

・引 き 数　　MRDINFO

	メンバ	型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixel_data	unsigned int	(-) 処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixel_bit	unsigned char	(-) 処理中半端画素データ・ビット数
	pixel_num	unsigned short	(i) １ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*pnint_buf	unsigned int	(-) プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(io) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	zero_cnt	unsigned short	(-) ０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(io) a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-) b1ポジション（〃）
	run_M_len	unsigned short	(io) メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-) パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-) 水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(io) 走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(io) ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-) 色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(io) 中断ステータス（０受信バッファ，１：変化点テーブル）

備考　ｉ：入力，ｏ：出力，io：入出力，-：未使用

保守／廃止

・返り値

返り値	説　　明
00	正常終了
01	中断
04	先頭にEOLを検出
06	１ライン分の画素数に満たないデータのあと，EOLを検出
07	１ライン分の画素数以上に復号化するデータが存在[注1]
08	非圧縮モード[注2]
09	異常なコードを検出[注2]

注１．たとえば，１ライン1728画素にて，1725画素まで復号化して次のデータを復号化した結果が５画素（計1730画素）だった場合にエラーとなります。1728画素まで復号した場合は次の符号データを復号化しません。

２．次のパラメータは，非圧縮モード，または異常なコード・データの直前に設定されます。

処理中半端符号データ・ビット数

処理中半端符号データ

受信バッファ残りサイズ

受信バッファ・アドレス

走査ライン用変化点テーブル・アドレス

変化点テーブル・バッファ残りサイズ

・機　　能　　指定されたメモリ上（受信バッファ）の符号データを入力し，MH復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

このライブラリでは，受信バッファ残りサイズが０のとき，または受信バッファ残りサイズが４，さらに処理中半端符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。このとき，中断ステータスは０になります。

１ライン処理中に変化点テーブル・バッファ残りサイズが０になると中断します。このとき，中断ステータスは１になります。変化点テーブル・バッファ残りサイズの指定はハーフワードでアラインした値が必要になります。変化点テーブル・バッファ残りサイズに０が入力された場合は10000Hが設定されます。

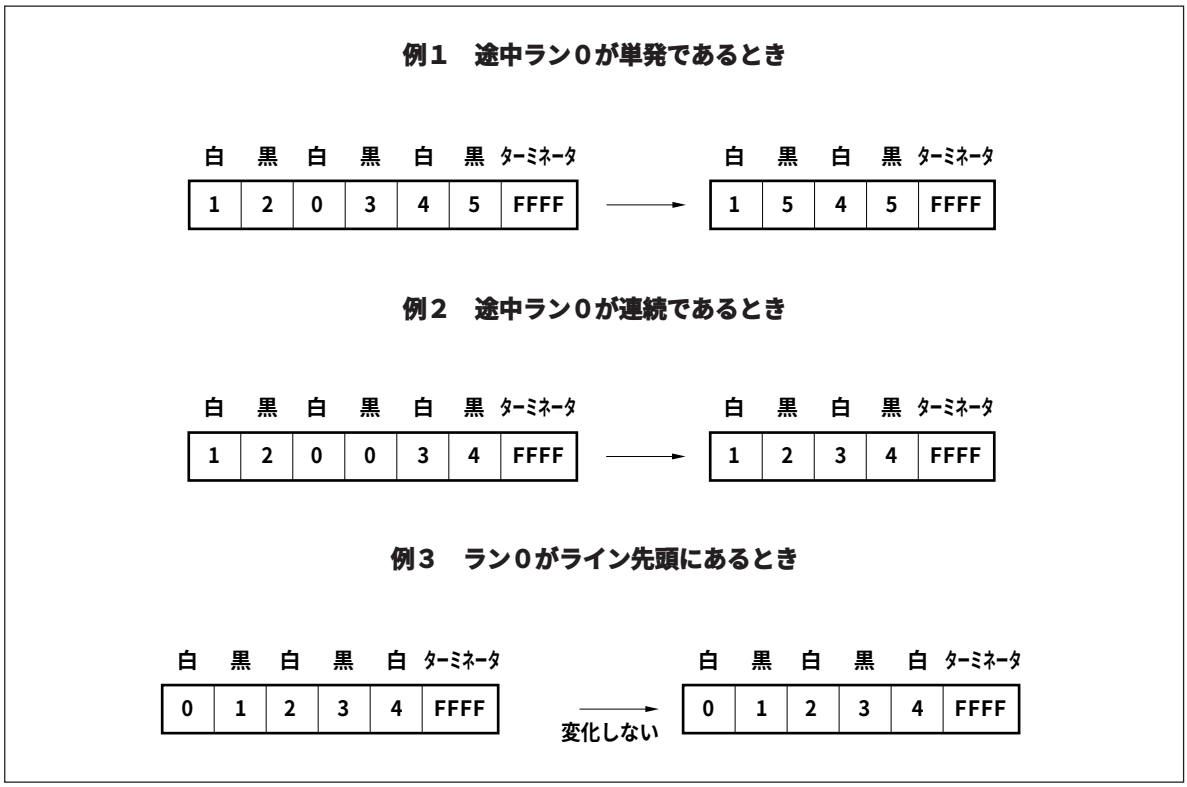
また，１ライン復号化処理中に途中ラン長０（ライン先頭を除く）を検出した場合，変化点テーブルを再構築します。

中断時に，内部情報を出力しますが，異常コード検出時にも内部情報のa0ポジション，メークアップ・ラン長，色（白黒）情報を出力します。

初期化のとき，再開アドレス／初期化フラグを０にしてください。

保守／廃止

図２－11　変化点テーブル再構築処理の概要（AP30100-B01）



保守／廃止

（８）MR復号化

・分　　類　　伸長処理系

・関 数 名　　mr_mrdec（）

・機能概要　　指定されたメモリ上（受信バッファ）の符号データおよび参照ライン用変化点情報（変化点テーブル）を入力し，MR復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

・形　　式　　#include ” codec_vr.h ”
int mr_mrdec（MRDINFO *mrdinfo）

・引 き 数　　MRDINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixel_data	unsigned int	(-)　処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixel_bit	unsigned char	(-)　処理中半端画素データ・ビット数
	pixel_num	unsigned short	(i)　１ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*print_buf	unsigned int	(-)　プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(io) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(io) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	zero_cnt	unsigned short	(-)　０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(io) a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(io) b1ポジション（〃）
	run_M_len	unsigned short	(io) メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(io) パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(io) 水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(io) 走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(io) ラン長０検出フラグ（〃）
	a0_color	unsigned char	(io) 色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(o)　中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断
04	先頭にEOLを検出
06	１ライン分の画素数に満たないデータのあと，EOLを検出
07	１ライン分の画素数以上に復号化するデータが存在[注1]
08	非圧縮モード[注2]
09	異常なコードを検出[注2]
10	ラン長でマイナスを検出
11	水平モード以外でラン長０を検出

注１．たとえば，１ライン1728画素で，1725画素まで復号化して次の符号データを復号化した結果が５画素（計1730画素）だった場合にエラーとなります。1728画素まで復号化した場合は次の符号データを復号化しません。

２．次のパラメータは，非圧縮モード，または異常なコード・データの直前に設定されます。

処理中半端符号データ・ビット数
処理中半端符号データ
受信バッファ残りサイズ
受信バッファ・アドレス
走査ライン用変化点テーブル・アドレス
参照ライン用変化点テーブル・アドレス
変化点テーブル・バッファ残りサイズ

・機　　能　　指定されたメモリ上（受信バッファ）の符号データおよび参照ライン用変化点情報（変化点テーブル）を入力し，MR復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

このライブラリでは，受信バッファ残りサイズが０のとき，または受信バッファ残りサイズが４，さらに処理中半端符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。このとき中断ステータスは０になります。

１ライン処理中に変化点テーブル・バッファ残りサイズが０になると中断します。このとき中断ステータスは１になります。変化点テーブル・バッファ残りサイズの指定はハーフワードでアラインした値が必要になります。変化点テーブル・バッファ残りサイズに０が入力された場合は10000Hが設定されます。

また，１ライン復号化処理中に途中ラン長０（ライン先頭を除く）を検出した場合，変化点テーブルを再構築します。

保守／廃止

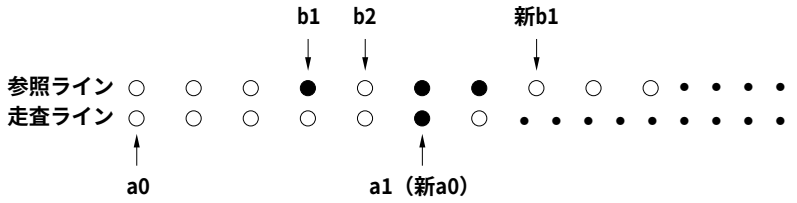
中断時に内部情報を出力しますが，異常コード検出時にも次の内部情報を出力します。

a0ポジション
b1ポジション
メークアップ・ラン長
パス・モード時のラン長
水平モード時の変化点テーブル要素数
色（白黒）情報

初期化のとき，再開アドレス／初期化フラグを０に設定してください。

★ **・不正なコードの処理**　本来はパス・コードとして処理されるべきものが，垂直コード：VR2，VR3として処理された場合の符号でも，新b1の位置を，新a0より右方向で，最初に現れる反対の画素位置へセットして，復号化処理を行います。
次に，不正なコードの処理例を示します。

図２－12　不正なコードの処理（AP30100-B01）



a0　各ラインを符号化する際に，ラインの最初の画素の直前に仮想的な白画素があるものとして，ここに置かれます。a0を開始点として１ラインのある範囲の符号化が終了すると，符号化された領域の後ろに新たにa0の位置が決定されます。
a1　符号化するライン上で，a0の次に現れる変化点に置かれます。
a2　符号化されるライン上で，a1の次に現れる変化点に置かれます。
b1　参照ライン上で，a0の右に現れる，a0の画素の色と反対色の最初の画素に置かれます。
b2　参照ライン上で，b1の次に現れる変化点に置かれます。

保守／廃止

（９）画素データ作成

・分　　類　　伸長処理系

・関 数 名　　mr_cnvtrp（）

・機能概要　　指定されたメモリ上（変化点テーブル）の１ライン分の変化点情報を入力し，画素データに変換し，指定されたメモリ上（プリント・バッファ）に送出します。

・形　　式　　#include ” codec_vr.h ”
int mr_cnvtrp（MRDINFO *mrdinfo）

・引 き 数　　MRDINFO

	メンバ	型	説　　明
入出力パラメータ	odd_data	unsigned int	(-)　処理中半端符号データ
	pixel_data	unsigned int	(io) 処理中半端画素データ
	odd_bit	unsigned char	(-)　処理中半端符号データ・ビット数
	pixel_bit	unsigned char	(io) 処理中半端画素データ・ビット数
	pixel_num	unsigned short	(i)　１ラインの画素数
	dec_buf_size	unsigned int	(-)　受信バッファ残りサイズ
	*dec_buf	unsigned int	(-)　受信バッファ・アドレス
	*print_buf	unsigned int	(io) プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(-)　再開アドレス／初期化フラグ
	zero_cnt	unsigned short	(-)　０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(-)　a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	run_M_len	unsigned short	(-)　メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-)　パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-)　水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(-)　走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(-)　ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(-)　中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に復号する変化点情報が存在

・機　　能　　指定されたメモリ上（変化点テーブル）の１ライン分の変化点情報を入力し，画素データに変換し，指定されたメモリ上（プリント・バッファ）に送出します。
１ライン分の画素数が32の倍数でない場合，データはプリント・バッファへ送出されずに処理中半端画素データ（ビット数は処理中半端画素データ・ビット数）として保持されます。中断処理はありません。

保守／廃止

（10）API符号化処理

・分　　類　　圧縮処理系（APIライブラリ使用）

・関 数 名　　mr_encode（）

・機能概要　　指定されたメモリ上（画素バッファ）から指定ライン分の画素データを入力し，符号化を行い，指定されたメモリ上（送信バッファ）に送出します。

・形　　式　　#include ” codec_vr.h ”
int mr_encode（MREINFO *mreinfo, MRAPIINFO *mrapiinfo）

・引 き 数　　MREINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	（io）処理中半端符号データ
	send_data	unsigned int	（io）送出符号データ
	odd_bit	unsigned char	（io）処理中半端符号データ・ビット数
	send_bit	unsigned char	（io）送出符号データ・ビット数
	pixel_num	unsigned short	（i）１ラインの画素数[注]
	enc_buf_size	unsigned int	（io）送信バッファ残りサイズ[注]
	*enc_buf	unsigned int	（io）送信バッファ・アドレス[注]
	*pixel_buf	unsigned int	（io）画素データ・バッファ・アドレス[注]
	*ref_tbl	unsigned short	（io）参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	（io）走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	（io）変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	（io）再開アドレス／初期化フラグ
	bit_length	unsigned short	（io）ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unsigned short	（io）a0ポジション（〃）
	a1pos	unsigned short	（io）a1ポジション（〃）
	b0pos	unsigned short	（io）b0ポジション（〃）
	b1pos	unsigned short	（io）b1ポジション（〃）
	b2pos	unsigned short	（io）b2ポジション（〃）
	exp_length	unsigned short	（io）拡張コード時残りレングス（〃）
	a0_color	unsigned char	（io）色（白黒）情報（〃）
	bit_offset	unsigned char	（io）ビット・サーチ・ワード内オフセット（〃）

注　ユーザ設定が必要なメンバです。それ以外のメンバは，ライブラリ内部で使用します。

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・引 き 数　　MRAPIINFO

メンバ		型	説　　明
入出力パラメータ	page_line	unsigned int	(i)　１ページのライン数
	pixel_line	unsigned int	(io) 処理ライン数
	line_cnt	unsigned int	(io) 累積ライン数カウンタ
	*run_len_tbl	unsigned short	(i)　変化点テーブル先頭アドレス
	reset	unsigned char	(io) １：リセット実行
	method	unsigned char	(i)　方式（１：MH, ２：MR, ３：MMR）
	k_para	unsigned char	(i)　Kパラメータ
	rtc_cnt	unsigned char	(io) RTCのEOLカウンタ
	fill_bit	unsigned int	(i)　最小送出ビット数
	restart_adr	unsigned int	(io) 再開アドレス
	mr_val［5］	unsigned int	(io) 変数退避エリア（中断時専用）

備考　i：入力，o：出力，io：入力，-：未使用

・返 り 値

返り値	説　　明
00	正常終了（指定ライン終了）
01	中断
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に復号する変化点情報が存在
13	ページ終了

・機　　能　　指定されたメモリ上（画素バッファ*pixel_buf）の指定ライン分（pixel_line）の画素データを入力して符号化を行い，指定された送信バッファ（*enc_buf）に送出します。送信バッファ残りサイズ（enc_buf_size）が０になると中断します。

（a）ページの先頭

ページの先頭で，リセット（reset = 1）してください。累積ライン数カウンタ（line_cnt）がクリアされます。

（b）ページの終了

累積ライン数カウンタ（line_cnt）が１ページのライン数（page_line）になると，RTC符号を付加して（あるいは付加せずに）ページ終了します。１ページのライン数があらかじめわからない場合は，FFFF FFFFHを設定し，わかったときにライン数を設定してください。また，次の場合でもページ終了となります。

保守／廃止

・指定ライン終了時にページ終了になった場合は，RTC符号を付加して（あるいは付加せずに）ページ終了となります。
・指定ライン先頭で，現在の累積ライン数カウンタ（line_cnt）のライン数を１ページのライン数（page_line）に設定した場合は，次のラインの符号化を行わないで，RTC符号を付加して（あるいは付加せずに）ページ終了となります。

１ページのライン数（page_line）に，現在の累積ライン数カウンタ（line_cnt）のライン数より小さい値を設定した場合の動作は保証できません。

（ｃ）フィル・ビット

１ライン当たりの最小送出ビット数（fill_bit）を設定してください。１ラインのEOL＋符号のビット数が，最小送出ビット数（fill_bit）より少ないときにフィル・ビットを付加します。MMR方式の場合は無効になります。

（ｄ）変化点テーブル

処理中に変化点テーブル・アドレス（*run_tbl, *ref_tbl）に対して変化点情報を入出力します。変化点テーブル残りサイズ（run_tbl_buf_size）に対しては，この関数内で変化点テーブルの最低限必要サイズを１ラインの画素数（pixel_num）から算出し設定します。また，変化点テーブル・アドレス（*run_tbl, *ref_tbl）に対してもこの関数内で設定します。

変化点テーブル先頭アドレス（*run_len_tbl）に変化点テーブルの先頭アドレスを設定してください。使用するサイズは次に示すとおりで，方式により異なります。

方式	変化点テーブルの使用サイズ
MH	１ライン分使用
MR	２ライン分使用
MMR	２ライン分使用

注意　メモリ領域の確保は，ユーザ側で行ってください。

備考　１ライン当たりの使用サイズ＝（１ラインの画素数＋３）×２バイト

保守／廃止

（11）API復号化処理

・分　　類　　伸長処理系（APIライブラリ使用）

・関 数 名　　mr_decode（ ）

・機能概要　　指定されたメモリ上（受信バッファ）から符号データを入力し，復号化を行い，指定されたメモリ上（プリント・バッファ）に送出します。

・形　　式　　#include ” codec_vr.h ”
int mr_decode（MRDINFO *mrdinfo, MRAPIINFO *mrapiinfo）

・引 き 数　　MRDINFO

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixel_data	unsigned int	(io) 処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixel_bit	unsigned char	(io) 処理中半端画素データ・ビット数
	pixel_num	unsigned short	(i) １ラインの画素数注
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ注
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス注
	*print_buf	unsigned int	(io) プリント・バッファ・アドレス注
	*ref_tbl	unsigned short	(io) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(io) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	zero_cnt	unsigned short	(io) ０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(io) a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(io) b1ポジション（〃）
	run_M_len	unsigned short	(io) メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(io) パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(io) 水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(io) 走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(io) ラン長０検出フラグ（〃）
	a0_color	unsigned char	(io) 色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(o) 中断ステータス（０：受信バッファ，１：変化点テーブル）

注　ユーザ設定が必要なメンバです。それ以外のメンバは，ライブラリ内部で使用します。

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・引 き 数　　MRAPIINFO

メンバ		型	説　　明
入出力パラメータ	page_line	unsigned int	(-)　１ページのライン数
	pixel_line	unsigned int	(io) 処理ライン数
	line_cnt	unsigned int	(io) 累積ライン数カウンタ
	*run_len_tbl	unsigned short	(i)　変化点テーブル先頭アドレス
	reset	unsigned char	(io) １：リセット実行
	method	unsigned char	(i)　方向（１：MH, ２：MR, ３：MMR）
	k_para	unsigned char	(-)　Ｋパラメータ
	rtc_cnt	unsigned char	(o) RTCのEOLカウンタ
	fill_bit	unsigned int	(-)　最小送出ビット数
	restart_adr	unsigned int	(io) 再開アドレス
	mr_val [5]	unsigned int	(io) 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

返り値	説　　明
00	正常終了（指定ライン終了）
01	中断
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に復号する変化点情報が存在
06	１ライン分の画素数に満たないデータのあと，EOLを検出
07	１ライン分の画素数以上に復号するデータが存在
08	非圧縮モード
09	異常なコードを検出
10	ラン・レングスがマイナスを検出
11	水平モード以上でランレングス“０”を検出
12	EOL以外のコードを検出した後，EOLを検出
13	ページ終了

・機　　能　　指定された受信バッファ（*dec_buf）から符号データを入力し，復号化を行い，指定されたプリント・バッファ（*print_buf）に指定ライン分（pixel_line）送出します。受信バッファ残りサイズ（dec_buf_size）が０になると中断します。

（ａ）ページの先頭

ページの先頭で，リセット（reset = 1）してください。累積ライン数カウンタ（line_cnt）がクリアされます。

保守／廃止

（ｂ）ページの終了

処理中にRTCを検出するとページ終了します。

（ｃ）非圧縮モード

非圧縮コードを検出した場合は中断します。再開する場合には，受信バッファ・アドレス（*enc_ buf）を次のラインの先頭の符号に設定してください。また，MR/MMR方式の場合は，検出したラインの走査ライン用変化点テーブル（*run_tbl）を作成してください。次のラインの参照ラインとして使用します。

（ｄ）エラー検出

エラーが発生した場合は中断します。再開できません。

（ｅ）変化点テーブル

処理中に変化点テーブル・アドレス（*run_tbl, *ref_tbl）に対して変化点情報を入出力します。変化点テーブル残りサイズ（run_tbl_buf_size）に対しては，この関数内で変化点テーブルの最低限必要サイズを１ラインの画素数（pixel_num）から算出し設定します。また，変化点テーブル・アドレス（*run_tbl, *ref_tbl）に対してもこの関数内で設定します。

変化点テーブル先頭アドレス（*run_len_tbl）に変化点テーブルの先頭アドレスを設定してください。使用するサイズは次に示すとおりで，方式により異なります。

方式	変化点テーブルの使用サイズ
MH	１ライン分使用
MR	２ライン分使用
MMR	２ライン分使用

注意　メモリ領域の確保は，ユーザ側で行ってください。

備考　１ライン当たりの使用サイズ＝（１ラインの画素数＋３）×２バイト

保守／廃止

２.３　関数仕様（AP703000-B01, AP705100-B01, AP70732-B01）

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。

入出力パラメータ＋内部情報領域は，圧縮／伸長系で次の構造体で定義されます。ヘッダ・ファイル“ codec810.h ”，“ codec830.h ”または“ codec850.h ”を参照してください。

２.３.１　AP703000-B01, AP705100-B01, AP70732-B01の構造体（パラメータ）

MH/MR/MMRの圧縮／伸長の関数で使用しているパラメータ（ENCODER, DECODER）を次に説明します。

（1）圧 縮 系

領域：94バイト

パラメータ：ENCODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	処理中半端符号データ
	send_data	unsigned int	送出符号データ
	odd_bit	unsigned char	処理中半端符号データ・ビット数
	send_bit	unsigned char	送出符号データ・ビット数
	pixcel_num	unsigned short	１ラインの画素数
	enc_buf_size	unsigned int	送信バッファ残りサイズ
	*enc_buf	unsigned int	送信バッファ・アドレス
	*pixcel_buf	unsigned int	画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	再開アドレス／初期化フラグ
	reg_area［10］	unsigned int	レジスタ退避エリア
	bit_length	unsigned short	ビット・サーチ残りレングス（中断処理時専門）
	a0pos	unsigned short	a0ポジション（〃）
	a1pos	unsigned short	a1ポジション（〃）
	b0pos	unsigned short	b0ポジション（〃）
	b1pos	unsigned short	b1ポジション（〃）
	b2pos	unsigned short	b2ポジション（〃）
	a0_color	unsigned char	色（白黒）情報（〃）
	bit_offset	unsigned char	ビット・サーチ・ワード内オフセット（〃）

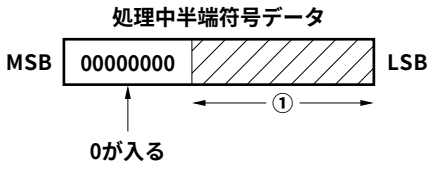
各メンバについて次に説明します。

・**odd_data**

符号化の終了した符号データで32ビットに満たない部分を，処理中半端符号データとしてLSBから格納します。詳細は**２．１．２（５）32ビットに満たない符号データの取り扱い**を参照してください。

符号データのデータ長は処理中半端符号データ・ビット数（odd_bit）となります。

図２－13　圧縮系の処理中半端符号データとデータ・ビット数の関係
（AP703000-B01, AP705100-B01, AP70732-B01）



①：処理中半端符号データ・ビット数（符号データ長）

・**send_data**

EOLやFILLなど，ユーザが符号データを送出するときに設定します。符号データ長は32ビット以下です。

処理中半端符号データ（odd_data）と合わせます。

詳細は**２．１．２（５）32ビットに満たない符号データの取り扱い**を参照してください。

・**odd_bit**

処理中半端符号データ（odd_data）中の処理中半端符号データ・ビット数（符号化の終了した符号データ・ビット数）を示します。

・**send_bit**

送出符号データ（send_data）の送出符号データ・ビット数を示します。

・**pixcel_num**

１ラインの画素数を示します。最大5183まで指定できます。

・**enc_buf_size**

送信バッファの残りサイズを示します。

処理中半端符号データが32ビット以上になり，送信バッファへ書き込まれるとデクリメント（－４）されます。このメンバが０になると処理を中断します。

このメンバには，ワードでアラインした値を入力してください。

・*enc_buf

送信バッファのアドレスを示します。

処理中半端符号データが32ビット以上になり，送信バッファへ書き込まれるとインクリメント（＋４）されます。

このメンバには，ワードでアラインした値を入力してください。

・*pixcel_buf

画像データのアドレスを示します。

画像データが32画素分圧縮されるとインクリメント（＋４）されます。

このメンバには，ワードでアラインした値を入力してください。

・*ref_tbl

参照ライン用の変化点テーブルのアドレスを示します。

このメンバには，ハーフワードでアラインした値を入力してください。

・*run_tbl

走査ライン用の変化点テーブルのアドレスを示します。

画素データがラン長に変換された場合などにインクリメント（＋２）されます。

このメンバには，ハーフワードでアラインした値を入力してください。

・run_tbl_buf_size

変化点テーブル・バッファ残りサイズを示します。

mr_coltrp（）においてデクリメント（－２）されます。

サイズはハーフワードでアラインした値で指定してください。０が入力された場合は10000Hが設定されます。

１ライン処理中にこのパラメータが０になると，処理を中断します。

・restart_adr

各ライブラリの処理中に中断した場合の再開アドレスを示します。

正常終了のときは０を返します。

中断情報を無視し，ライブラリの処理を初期化したいときは０を設定してください。

・reg_area［10］

レジスタ退避エリアに使用します。

・bit_length

mr_coltrp（）において中断が発生したとき，１ラインの残り画素データをビット・サーチする残りレングスを示します。

保守／廃止

• **a0pos**

mr_mhenc（），mr_mrenc（）において中断が発生したときのa0ポジションを示します。

• **a1pos**

mr_mrenc（）において中断が発生したときのa1ポジションを示します。

• **b0pos**

mr_mrenc（）において中断が発生したときのb0ポジションを示します。

• **b1pos**

mr_mrenc（）において中断が発生したときのb1ポジションを示します。

• **b2pos**

mr_mrenc（）において中断が発生したときのb2ポジションを示します。

• **a0_color**

mr_mrenc（）において中断が発生したときのa0ポジションの色（白黒）情報を示します（0H：白，FFH：黒）。

• **bit_offset**

mr_coltrp（）において中断が発生したとき，画素データをビット・サーチしたワード内のオフセット値を示します。

（2）伸 長 系

領域：92バイト（AP705100-B01, AP70732-B01）

96バイト（AP703000-B01）

パラメータ：DECODER

	メンバ	型	説　　明
入出力パラメータ	odd_data	unsigned int	処理中半端符号データ
	pixcel_data	unsigned int	処理中半端画素データ
	odd_bit	unsigned char	処理中半端符号データ・ビット数
	pixcel_bit	unsigned char	処理中半端画素データ・ビット数
	pixcel_num	unsigned short	１ラインの画素数
	dec_buf_size	unsigned int	受信バッファ残りサイズ
	*dec_buf	unsigned int	受信バッファ・アドレス
	*print_buf	unsigned int	プリント・バッファ・アドレス
	*ref_tbl	unsigned short	参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	再開アドレス／初期化フラグ
	reg_area［8］	unsigned int	レジスタ退避エリア（AP705100-B01, AP70732-B01の場合です。AP703000-B01の場合はreg_area［9］です）
	zero_cnt	unsigned short	０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	b1ポジション（〃）
	run_M_len	unsigned short	メークアップ・ラン長（〃）
	run_pass_len	unsigned short	パス・モード時のラン長（〃）
	chg_cnt	unsigned short	水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	ラン長０検出フラグ（〃）
	a0_color	unsigned char	色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	中断ステータス

各メンバについて次に説明します。

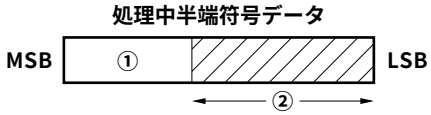
・**odd_data**

伸長対象となる32ビットの符号データを，受信バッファから処理中半端符号データ（odd_data）にLSBから格納し保持します。伸長処理が終了した符号データ長は処理中半端符号データ・ビット数（odd_bit）に保持します。

保守／廃止

図２－14　伸長系の処理中半端符号データとデータ・ビット数の関係

（AP703000-B01, AP705100-B01, AP70732-B01）



①：まだ復号していないデータ

②：処理中半端符号データ・ビット数（復号が終わったデータ長）

・**pixcel_data**

伸長処理後の32ビットに満たない画素データを保持します。

・**odd_bit**

処理中半端符号データ（odd_data）中の処理中半端符号データ・ビット数（伸長処理が終了した符号データ・ビット数）を示します。

・**pixcel_bit**

処理中半端画素データ（pixcel_data）中の画素データ・ビット数を示します。

・**pixcel_num**

１ラインの画素数を示します。

・**dec_buf_size**

受信バッファの残りサイズを示します。

受信バッファから符号データを読み出すとデクリメント（－４）されます。このメンバが０のとき，またはこのメンバが４，さらに処理中半端符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。

このメンバには，ワードでアラインした値を入力してください。

・***dec_buf**

受信バッファのアドレスを示します。

処理中半端符号データへ送出するとインクリメント（＋４）されます。

このメンバには，ワードでアラインした値を入力してください。

・***print_buf**

画像データのアドレスを示します。

画像データが32画素分伸長されるとインクリメント（＋４）されます。

このメンバには，ワードでアラインした値を入力してください。

保守／廃止

・*ref_tbl

参照ライン用の変化点テーブルのアドレスを示します。

このメンバには，ハーフワードでアラインした値を入力してください。

・*run_tbl

走査ライン用の変化点テーブルのアドレスを示します。

符号データがラン長に変換された場合などにインクリメント（＋２）されます。

このメンバには，ハーフワードでアラインした値を入力してください。

・run_tbl_buf_size

変化点テーブル・バッファ残りサイズを示します。

mr_mhdec（），mr_mrdec（）においてデクリメント（－２）されます。

サイズはハーフワードでアラインした値で指定してください。０が入力された場合は10000Hが設定されます。

１ライン処理中にこのパラメータが０になると，処理を中断します。

・restart_adr

各ライブラリの処理中に中断した場合の再開アドレスを示します。

初期化のときは，０に設定してください。

・reg_area［8］注

レジスタ退避エリアに使用します。

注　AP705100-B01, AP70732-B01の場合です。AP703000-B01の場合はreg_area［9］です。

・zero_cnt

mr_scheol（）において中断が発生したときのEOLコードの０の数を保持します。

内部では32ビット・レジスタを使用していますが，入出力のパラメータとしては上位16ビットをマスクして出力しています。ただし，動作上は保証しています。

・a0pos

mr_mhdec（），mr_mrdec（）において，中断発生時および異常コード検出時のa0ポジションを示します。

・b1pos

mr_mrdec（）において，中断発生時および異常コード検出時のb1ポジションを示します。

保守／廃止

・run_M_len

mr_mhdec（），mr_mrdec（）において，メークアップ・コード復号後の中断発生時および異常コード検出時のメークアップ・コードのラン長を示します。

・run_pass_len

mr_mrdec（）において，パス・モード復号後の中断発生時および異常コード検出時のパス・モードのラン長を示します。

・chg_cnt

mr_mrdec（）において，水平モード復号中の中断発生時および異常コード検出時の水平モードの変化点テーブル要素数を示します。

・run_tbl_sadr

mr_mhdec（），mr_mrdec（）において，中断が発生したときの走査ライン用変化点テーブルの先頭アドレスを示します。

・run0_flag

mr_mhdec（），mr_mrdec（）において，中断が発生したときの変化点テーブル再構築用のラン長０検出フラグを示します（０：ラン長０データなし，１：ラン長０データあり）。

・a0_color

mr_mrdec（）において，中断発生時および異常コード検出時のa0ポジションの色（白，黒）情報を示します（0H：白，FFH：黒）。

・int_sts

mr_mhdec（），mr_mrdec（）において，中断が発生した場合の中断の原因を示します。

０：受信バッファ残りサイズがない（dec_buf_size = 0 or(dec_buf_size = 4 and odd_bit＜13)）

１：変化点テーブル・バッファ残りサイズがない（run_tbl_buf_size = 0）

保守／廃止

2.3.2　外部インタフェース（AP703000-B01, AP705100-B01, AP70732-B01）

（1）データ送出

・分　　類　　圧縮処理系

・関 数 名　　mr_albit（）

・機能概要　　EOL, FILLなど32ビット以下のデータを送信バッファへ出力します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_albit（struct ENCODER *encdata）

・引 き 数　　ENCODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	send_data	unsigned int	(i)　送出符号データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	send_bit	unsigned char	(i)　送出符号データ・ビット数
	pixcel_num	unsigned short	(-)　１ラインの画素数
	enc_buf_size	unsigned int	(io) 送信バッファ残りサイズ
	*enc_buf	unsigned int	(io) 送信バッファ・アドレス
	*pixcel_buf	unsigned int	(-)　画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(-)　走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(-)　再開アドレス／初期化フラグ
	reg_area［10］	unsigned int	(io) レジスタ退避エリア
	bit_length	unsigned short	(-)　ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unsigned short	(-)　a0ポジション（〃）
	a1pos	unsigned short	(-)　a1ポジション（〃）
	b0pos	unsigned short	(-)　b0ポジション（〃）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	b2pos	unsigned short	(-)　b2ポジション（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（〃）
	bit_offset	unsigned char	(-)　ビット・サーチ・ワード内オフセット（〃）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断

・機　　能　　EOL, FILLなど32ビット以下のデータを送信バッファへ出力する処理を行います。送信バッファ残りサイズが０になると中断します。
処理中半端符号データについては２.１.２（５）32ビットに満たない符号データの取り扱いを参照してください。

注意　送信バッファの残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください（入力できる値については２.３.１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（２）変化点テーブル作成

・分　　類　　圧縮処理系

・関 数 名　　mr_coltrp（）

・機能概要　　１ライン分の画素データの変化点テーブルを指定されたメモリ上に生成します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_coltrp（struct ENCODER *encdata）

・引 き 数　　ENCODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(-)　処理中半端符号データ
	send_data	unsigned int	(-)　送出符号データ
	odd_bit	unsigned char	(-)　処理中半端符号データ・ビット数
	send_bit	unsigned char	(-)　送出符号データ・ビット数
	pixcel_num	unsigned short	(i)　１ラインの画素数
	enc_buf_size	unsigned int	(-)　送信バッファ残りサイズ
	*enc_buf	unsigned int	(-)　送信バッファ・アドレス
	*pixcel_buf	unsigned int	(io) 画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(io) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	reg_area［10］	unsigned int	(io) レジスタ退避エリア
	bit_length	unsigned short	(io) ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unsigned short	(-)　a0ポジション（〃）
	a1pos	unsigned short	(-)　a1ポジション（〃）
	b0pos	unsigned short	(-)　b0ポジション（〃）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	b2pos	unsigned short	(-)　b2ポジション（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（〃）
	bit_offset	unsigned char	(io) ビット・サーチ・ワード内オフセット（〃）

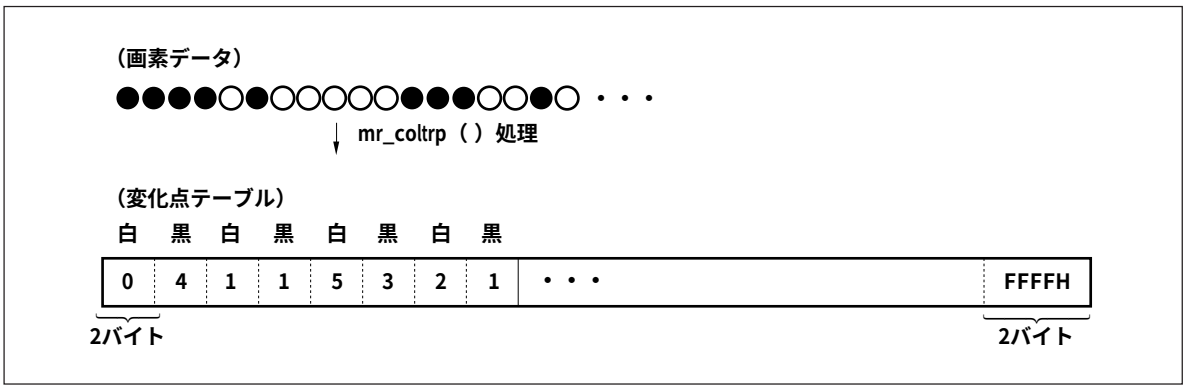
備考　i：入力，o：出力，io：入出力，-：未使用

・返 り 値

返り値	説　　明
00	正常終了
01	中断（変換点テーブル・サイズ・オーバフロー）

・機　　能　　符号化処理において，１ライン分の画素データの変化点テーブルを指定されたメモリ上に生成します。１ライン処理中に変化点テーブル残りサイズが０になると中断します。変化点テーブル・サイズの指定はハーフワードでアラインした値が必要になります。変化点テーブル・サイズに０が入力された場合は10000Hが設定されます。
変化点テーブルは白のラン・レングスを先頭に２バイトごとに各色（白／黒）のラン・レングスを順に格納します。変化点テーブルの最終２バイトには１ライン分の終了情報として“ FFFFH ”を格納します。初期化のときに再開アドレス／初期化フラグを０に設定してください。図２－15に変化点テーブルのフォーマットを示します。

図２－15　変化点テーブルのフォーマット（AP703000-B01, AP705100-B01, AP70732-B01）



１ライン処理で中断させないためには，変化点テーブル領域として（１ラインの画素数＋１）×２バイトのメモリを確保してください。

保守／廃止

（３）MH符号化

・分　　類　　圧縮処理系

・関 数 名　　mr_mhenc（）

・機能概要　　１ライン（走査ライン）分の画素データの変化点情報（変化点テーブル）を入力し，MH符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_mhenc（struct ENCODER *encdata）

・引 き 数　　ENCODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	send_data	unsigned int	(-)　送出符号データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	send_bit	unsigned char	(-)　送出符号データ・ビット数
	pixcel_num	unsigned short	(i)　１ラインの画素数
	enc_buf_size	unsigned int	(io) 送信バッファ残りサイズ
	*enc_buf	unsigned int	(io) 送信バッファ・アドレス
	*pixcel_buf	unsigned int	(-)　画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	reg_area［10］	unsigned int	(io) レジスタ退避エリア
	bit_length	unsigned short	(-)　ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unsigned short	(io) a0ポジション（〃）
	a1pos	unsigned short	(-)　a1ポジション（〃）
	b0pos	unsigned short	(-)　b0ポジション（〃）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	b2pos	unsigned short	(-)　b2ポジション（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（〃）
	bit_offset	unsigned char	(-)　ビット・サーチ・ワード内オフセット（〃）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に符号化する変化点情報が存在

・機　　能　　１ライン（走査ライン）分の画素データの変化点情報（変化点テーブル）を入力し，MH符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。１ライン符号化の終了条件は，変化点テーブルのターミネータ（FFFFH）の検出時です。１ライン処理中に送信バッファ残りサイズが０になると中断します。初期化のときに再開アドレス／初期化フラグを０に設定してください。

注意　送信バッファの残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください（入力できる値については２.３.１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（4）MR符号化

・分　　類　　圧縮処理系

・関 数 名　　mr_mrenc（）

・機能概要　　参照ライン／走査ラインの変化点情報（変化点テーブル）を入力し，１ライン分のMR符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_mrenc（struct ENCODER *encdata）

・引 き 数　　ENCODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	send_data	unsigned int	(-)　送出符号データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	send_bit	unsigned char	(-)　送出符号データ・ビット数
	pixcel_num	unsigned short	(i)　１ラインの画素数
	enc_buf_size	unsigned int	(io) 送信バッファ残りサイズ
	*enc_buf	unsigned int	(io) 送信バッファ・アドレス
	*pixcel_buf	unsigned int	(-)　画素データ・バッファ・アドレス
	*ref_tbl	unsigned short	(io) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	reg_area［10］	unsigned int	(io) レジスタ退避エリア
	bit_length	unsigned short	(-)　ビット・サーチ残りレングス（中断処理時専用）
	a0pos	unsigned short	(io) a0ポジション（〃）
	a1pos	unsigned short	(io) a1ポジション（〃）
	b0pos	unsigned short	(io) b0ポジション（〃）
	b1pos	unsigned short	(io) b1ポジション（〃）
	b2pos	unsigned short	(io) b2ポジション（〃）
	a0_color	unsigned char	(io) 色（白黒）情報（〃）
	bit_offset	unsigned char	(-)　ビット・サーチ・ワード内オフセット（〃）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に符号化する変化点情報が存在

・機　　能　　参照ライン／走査ラインの変化点情報（変化点テーブル）を入力し，１ライン分のMR符号化を行い，その符号を指定されたメモリ上（送信バッファ）に送出します。１ライン符号化の終了条件は，走査用変化点テーブルのターミネータ（FFFFH）の検出時です。

１ライン処理中に送信バッファ残りサイズが０になると中断します。初期化のときに再開アドレス／初期化フラグを０に設定してください。

注意　送信バッファの残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください（入力できる値については２.３.１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（５）EOL検出

・分　　類　　伸長処理系

・関 数 名　　mr_scheol（）

・機能概要　　メモリ上（受信バッファ）の指定された位置からEOLを検出します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_scheol（struct DECODER *decdata）

・引 き 数　　DECODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixcel_data	unsigned int	(-)　処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixcel_bit	unsigned char	(-)　処理中半端画素データ・ビット数
	pixcel_num	unsigned short	(-)　１ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*print_buf	unsigned int	(-)　プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(-)　走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	reg_area［8］	unsigned int	(io) レジスタ退避エリア
	zero_cnt	unsigned short	(io) ０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(-)　a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	run_M_len	unsigned short	(-)　メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-)　パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-)　水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(-)　走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(-)　ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(-)　中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
01	中断
04	先頭にEOLを検出（正常終了）
05	FILL付きEOLを先頭に検出（正常終了）
12	EOL以外のコードを検出したあと，EOLを検出

・機　　能　　復号化処理において，メモリ上（受信バッファ）の指定された位置からEOLを検出します。指定された受信バッファ内にEOLが検出されない場合に中断します。初期化のとき，再開アドレス／初期化フラグを０に設定してください。
このライブラリでは，受信バッファ残りサイズが０のとき，または受信バッファ残りサイズが４，さらに処理中半端符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。

注意１．受信バッファの残りサイズが4Hで，処理中半端符号データ・ビット数が13ビット未満にEOLコードが存在しても中断します。
２．０ビット・カウンタは，内部では32ビット・レジスタを使用していますが，入出力パラメータの出力としては上位16ビットをマスクして出力しています。ただし，動作上は保証しています。

保守／廃止

（６）１ビット検出

・分　　類　　伸長処理系

・関 数 名　　mr_getbit（）

・機能概要　　タグ・ビットなどの１ビットを検出します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_getbit（struct DECODER *decdata）

・引 き 数　　DECODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixcel_data	unsigned int	(-)　処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixcel_bit	unsigned char	(-)　処理中半端画素データ・ビット数
	pixcel_num	unsigned short	(-)　１ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*print_buf	unsigned int	(-)　プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(-)　走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(-)　再開アドレス／初期化フラグ
	reg_area［8］	unsigned int	(io) レジスタ退避エリア
	zero_cnt	unsigned short	(-)　０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(-)　a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	run_M_len	unsigned short	(-)　メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-)　パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-)　水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(-)　走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(-)　ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(-)　中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　i：入力，o：出力，io：入出力，-：未使用

・返 り 値

返り値	説　　明
00	タグ・ビットが０
01	タグ・ビットが１

・機　　能　　復号化処理において，タグ・ビットなどの１ビットを検出します。メモリ上（受信バッファ）の指定された位置から１ビットを取り出し，関数の返り値に設定します。

注意　受信バッファ残りサイズが0Hのとき，このライブラリを実行すると動作は保証されません。必ず4H以上の値を入力してから実行してください（入力できる値については２.３.１（１）圧縮系　enc_buf_sizeを参照）。

保守／廃止

（７）MH復号化

・分　　類　　伸長処理系

・関 数 名　　mr_mhdec（）

・機能概要　　指定されたメモリ上（受信バッファ）の符号データを入力し，MH復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_mhdec（struct DECODER *decdata）

・引 き 数　　DECODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixcel_data	unsigned int	(-)　処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixcel_bit	unsigned char	(-)　処理中半端画素データ・ビット数
	pixcel_num	unsigned short	(i)　１ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*print_buf	unsigned int	(-)　プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(io) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	reg_area［8］	unsigned int	(io) レジスタ退避エリア
	zero_cnt	unsigned short	(-)　０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(io) a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	run_M_len	unsigned short	(io) メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-)　パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-)　水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(io) 走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(io) ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(io) 中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断
04	先頭にEOLを検出
06	１ライン分の画素数に満たないデータのあと，EOLを検出
07	１ライン分の画素数以上に復号化するデータが存在[注1]
08	非圧縮モード[注2]
09	異常なコードを検出[注2]

注１．たとえば，１ライン1728画素で，1725画素まで復号化して次の符号データを復号化した結果が５画素（計1730画素）だった場合にエラーとなります。1728画素まで復号化した場合は次の符号データを復号化しません。

２．次のパラメータは，非圧縮モード，または異常なコード・データの直前に設定されます。

処理中半端符号データ・ビット数
処理中半端符号データ
受信バッファ残りサイズ
受信バッファ・アドレス
走査ライン用変化点テーブル・アドレス
変化点テーブル・バッファ残りサイズ

・機　　能　　指定されたメモリ上（受信バッファ）の符号データを入力し，MH復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

このライブラリでは，受信バッファ残りサイズが０のとき，または受信バッファ残りサイズが４，さらに処理中半端符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。このとき中断ステータスは０になります。

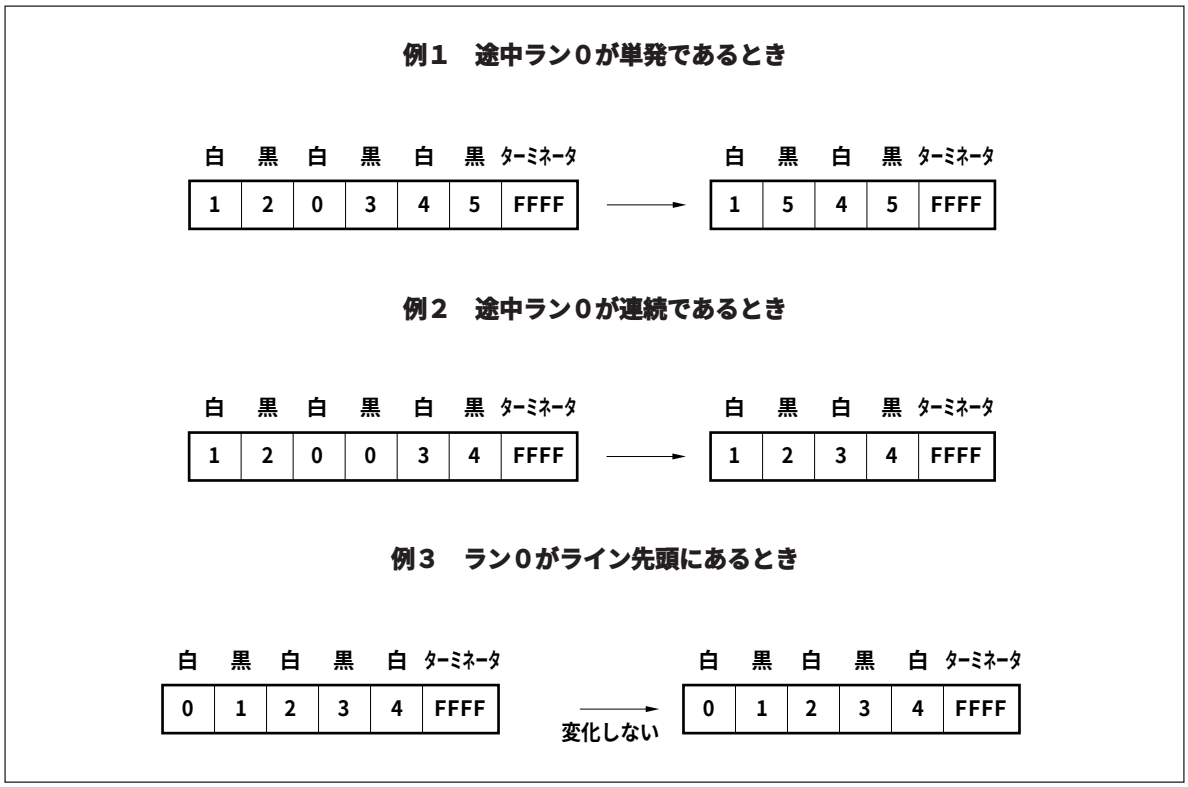
１ライン処理中に変化点テーブル・バッファ残りサイズが０になると中断します。このとき中断ステータスは１になります。変化点テーブル・バッファ残りサイズの指定はハーフワードでアラインした値が必要になります。変化点テーブル・バッファ残りサイズに０が入力された場合は10000Hが設定されます。

また，１ライン復号化処理中に途中ラン０（ライン先頭を除く）を検出した場合，変化点テーブルを再構築します。

中断時に内部情報を出力しますが，異常コード検出時にも内部情報のa0ポジション，メークアップ・ラン長，色（白黒）情報を出力します。

初期化のとき，再開アドレス／初期化フラグを０に設定してください。

図２－16　変化点テーブル再構築処理の概要（AP703000-B01, AP705100-B01, AP70732-B01）



保守／廃止

（８）MR復号化

・分　　類　　伸長処理系

・関 数 名　　mr_mrdec（）

・機能概要　　指定されたメモリ上（受信バッファ）の符号データおよび参照ライン用変化点情報（変化点テーブル）を入力し，MR復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_mrdec（struct DECODER *decdata）

・引 き 数　　DECODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(io) 処理中半端符号データ
	pixcel_data	unsigned int	(-)　処理中半端画素データ
	odd_bit	unsigned char	(io) 処理中半端符号データ・ビット数
	pixcel_bit	unsigned char	(-)　処理中半端画素データ・ビット数
	pixcel_num	unsigned short	(i)　１ラインの画素数
	dec_buf_size	unsigned int	(io) 受信バッファ残りサイズ
	*dec_buf	unsigned int	(io) 受信バッファ・アドレス
	*print_buf	unsigned int	(-)　プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(io) 参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(io) 変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(io) 再開アドレス／初期化フラグ
	reg_area [8]	unsigned int	(io) レジスタ退避エリア
	zero_cnt	unsigned short	(-)　０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(io) a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(io) b1ポジション（〃）
	run_M_len	unsigned short	(io) メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(io) パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(io) 水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(io) 走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(io) ラン長０検出フラグ（〃）
	a0_color	unsigned char	(io) 色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(o)　中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
01	中断
04	先頭にEOLを検出
06	１ライン分の画素数に満たないデータのあと，EOLを検出
07	１ライン分の画素数以上に復号化するデータが存在[注1]
08	非圧縮モード[注2]
09	異常なコードを検出[注2]
10	ラン長でマイナスを検出
11	水平モード以外でラン長０を検出

注１．たとえば，１ライン1728画素で，1725画素まで復号化して次の符号データを復号化した結果が５画素（計1730画素）だった場合にエラーとなります。1728画素まで復号化した場合は次の符号データを復号化しません。

２．次のパラメータは，非圧縮モード，または異常なコード・データの直前に設定されます。

処理中半端符号データ・ビット数
処理中半端符号データ
受信バッファ残りサイズ
受信バッファ・アドレス
走査ライン用変化点テーブル・アドレス
参照ライン用変化点テーブル・アドレス
変化点テーブル・バッファ残りサイズ

・機　　能　　指定されたメモリ上（受信バッファ）の符号データおよび参照ライン用変化点情報（変化点テーブル）を入力し，MR復号化を行い，生成された変化点情報を指定されたメモリ上（変化点テーブル）に送出します。

このライブラリでは，受信バッファ残りサイズが０のとき，または受信バッファ残りサイズが４，さらに処理中半端符号データ・ビット数が13ビット未満のときに中断します（符号コードの最大長が13ビットのため）。このとき中断ステータスは０になります。

１ライン処理中に変化点テーブル・バッファ残りサイズが０になると中断します。このとき中断ステータスは１になります。変化点テーブル・バッファ残りサイズの指定はハーフワードでアラインした値が必要になります。変化点テーブル・バッファ残りサイズに０が入力された場合は10000Hが設定されます。

また，１ライン復号化処理中に途中ラン長０（ライン先頭を除く）を検出した場合，変化点テーブルを再構築します。

保守／廃止

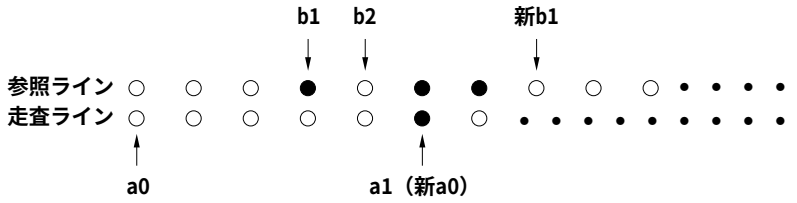
中断時に内部情報を出力しますが，異常コード検出時にも次の内部情報を出力します。

a0ポジション
b1ポジション
メークアップ・ラン長
パス・モード時のラン長
水平モード時の変化点テーブル要素数
色（白黒）情報

初期化のとき，再開アドレス／初期化フラグを０に設定してください。

★ **・不正なコードの処理**　本来はパス・コードとして処理されるべきものが，垂直コード：VR2，VR3として処理された場合の符号でも，新b1の位置を，新a0より右方向で，最初に現れる反対の画素位置へセットして，復号化処理を行います。
次に，不正なコードの処理例を示します。

図２－17　不正なコードの処理（AP703000-B01, AP705100-B01, AP70732-B01）



a0　各ラインを符号化する際に，ラインの最初の画素の直前に仮想的な白画素があるものとして，ここに置かれます。a0を開始点として１ラインのある範囲の符号化が終了すると，符号化された領域の後ろに新たにa0の位置が決定されます。
a1　符号化するライン上で，a0の次に現れる変化点に置かれます。
a2　符号化されるライン上で，a1の次に現れる変化点に置かれます。
b1　参照ライン上で，a0の右に現れる，a0の画素の色と反対色の最初の画素に置かれます。
b2　参照ライン上で，b1の次に現れる変化点に置かれます。

保守／廃止

（９）画素データ作成

・分　　類　　伸長処理系

・関 数 名　　mr_cnvtrp（）

・機能概要　　指定されたメモリ上（変化点テーブル）の１ライン分の変化点情報を入力し，画素データに変換し，指定されたメモリ上（プリント・バッファ）に送出します。

・形　　式　　#include ” codec810.h ”（V810ファミリの場合）
#include ” codec830.h ”（V830ファミリの場合）
#include ” codec850.h ”（V850ファミリの場合）
int mr_cnvtrp（struct DECODER *decdata）

・引 き 数　　DECODER

メンバ		型	説　　明
入出力パラメータ	odd_data	unsigned int	(-)　処理中半端符号データ
	pixcel_data	unsigned int	(io) 処理中半端画素データ
	odd_bit	unsigned char	(-)　処理中半端符号データ・ビット数
	pixcel_bit	unsigned char	(io) 処理中半端画素データ・ビット数
	pixcel_num	unsigned short	(i)　１ラインの画素数
	dec_buf_size	unsigned int	(-)　受信バッファ残りサイズ
	*dec_buf	unsigned int	(-)　受信バッファ・アドレス
	*print_buf	unsigned int	(io) プリント・バッファ・アドレス
	*ref_tbl	unsigned short	(-)　参照ライン用変化点テーブル・アドレス
	*run_tbl	unsigned short	(io) 走査ライン用変化点テーブル・アドレス
	run_tbl_buf_size	unsigned int	(-)　変化点テーブル・バッファ残りサイズ
内部領域	restart_adr	unsigned int	(-)　再開アドレス／初期化フラグ
	reg_area［8］	unsigned int	(io) レジスタ退避エリア
	zero_cnt	unsigned short	(-)　０ビット・カウンタ（中断処理専用）
	a0pos	unsigned short	(-)　a0ポジション（中断処理時，異常コード検出時）
	b1pos	unsigned short	(-)　b1ポジション（〃）
	run_M_len	unsigned short	(-)　メークアップ・ラン長（〃）
	run_pass_len	unsigned short	(-)　パス・モード時のラン長（〃）
	chg_cnt	unsigned short	(-)　水平モード時の変化点テーブル要素数（〃）
	run_tbl_sadr	unsigned int	(-)　走査ライン用変化テーブル先頭アドレス（中断処理専用）
	run0_flag	unsigned short	(-)　ラン長０検出フラグ（〃）
	a0_color	unsigned char	(-)　色（白黒）情報（中断処理時，異常コード検出時）
	int_sts	unsigned char	(-)　中断ステータス（０：受信バッファ，１：変化点テーブル）

備考　i：入力，o：出力，io：入出力，-：未使用

保守／廃止

・返 り 値

返り値	説　　明
00	正常終了
02	１ライン分の画素数に満たない変化点情報のあと，FFFFHを検出
03	１ライン分の画素数以上に復号する変化点情報が存在

・機　　能　　指定されたメモリ上（変化点テーブル）の１ライン分の変化点情報を入力し，画素データに変換し，指定されたメモリ上（プリント・バッファ）に送出します。
１ライン分の画素数が32の倍数でない場合，データはプリント・バッファへ送出されずに処理中半端画素データ（ビット数は処理中半端画素データ・ビット数）として保持されます。中断処理はありません。

〔メ　モ〕

保守／廃止

第３章　インストレーション

３.１　リンク方法

次にAP30100-B01, AP703000-B01, AP705100-B01, AP70732-B01で使用しているセクション名を示します。

セクション名	属性	機　能
.MRETEXT	text	MH/MR/MMR圧縮処理プログラム
.MREDATA	data	符号化テーブル
.MRDTEXT	text	MH/MR/MMR伸長処理プログラム
.MRDDATA	data	復号化テーブル

次に，リンク方法の概要をリンカごとに示します。

★　（１）AP30100-B01（VR4100シリーズ）

・GHS製　リンカ　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〉-lmhmr

（２）AP703000-B01（V850ファミリ）

・NEC製　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〉.../libmhmr.a -o〈出力ファイル〉

・GHS製　リンカ　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〉-lmhmr

（３）AP705100-B01（V830ファミリ）

・NEC製　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〉.../libmhmr.a -o〈出力ファイル〉

・GHS製　リンカ　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〉-lmhmr

保守／廃止

（４）AP70732-B01（V810ファミリ）

・NEC製　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〉.../libmhmr.a -o〈出力ファイル〉

・GHS製　リンカ　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〉-lmhmr

★ 3.2　サンプル・プログラムのリンク

3.2.1　VR4100シリーズ用サンプル

AP30100-B01のmakefileは次のようになっています。

（1）APIライブラリを使用しない場合（makenorm）

```
CC = ccmipel
AS = asmips
LD = lx

all:mh_enc.elf mr_enc.elf mmr_enc.elf mh_dec.elf mr_dec.elf mmr_dec.elf

mh_enc.elf: crt4100.o mh_enc.o makenorm
	$(LD) -o mh_enc.elf -e __start -sec { .text 0x80010000 : .MRETEXT : .data
0x84000000 : .MREDATA : .sdata : .sbss : .bss } crt4100.o mh_enc.o -L../../lib4100 -lmhmr -M

mr_enc.elf: crt4100.o mr_enc.o makenorm
	$(LD) -o mr_enc.elf -e __start -sec { .text 0x80010000 : .MRETEXT : .data
0x84000000 : .MREDATA : .sdata : .sbss : .bss } crt4100.o mr_enc.o -L../../lib4100 -lmhmr -M

mmr_enc.elf: crt4100.o mmr_enc.o makenorm
	$(LD) -o mmr_enc.elf -e __start -sec { .text 0x80010000 : .MRETEXT : .data
0x84000000 : .MREDATA : .sdata : .sbss : .bss } crt4100.o mmr_enc.o -L../../lib4100 -lmhmr -M

mh_dec.elf: crt4100.o mh_dec.o makenorm
	$(LD) -o mh_dec.elf -e __start -sec { .text 0x80010000 : .MRDTEXT : .data
0x84000000 : .MRDDATA : .sdata : .sbss : .bss } crt4100.o mh_dec.o -L../../lib4100 -lmhmr -M

mr_dec.elf: crt4100.o mr_dec.o makenorm
	$(LD) -o mr_dec.elf -e __start -sec { .text 0x80010000 : .MRDTEXT : .data
0x84000000 : .MRDDATA : .sdata : .sbss : .bss } crt4100.o mr_dec.o -L../../lib4100 -lmhmr -M

mmr_dec.elf: crt4100.o mmr_dec.o makenorm
	$(LD) -o mmr_dec.elf -e __start -sec { .text 0x80010000 : .MRDTEXT : .data
0x84000000 : .MRDDATA : .sdata : .sbss : .bss } crt4100.o mmr_dec.o -L../../lib4100 -lmhmr -M

mh_enc.o: mh_enc.c makenorm
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mh_enc.o mh_enc.c

mr_enc.o: mr_enc.c makenorm
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mr_enc.o mr_enc.c

mmr_enc.o: mmr_enc.c makenorm
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mmr_enc.o mmr_enc.c

mh_dec.o: mh_dec.c makenorm
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mh_dec.o mh_dec.c

mr_dec.o: mr_dec.c makenorm
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mr_dec.o mr_dec.c

mmr_dec.o: mmr_dec.c makenorm
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mmr_dec.o mmr_dec.c

crt4100.o: crt4100.s makenorm
	$(AS) -elf -r4100 -b0 -o crt4100.o crt4100.s
```

（２）APIライブラリを使用する場合（makeapi）

```
CC = ccmipel
AS = asmips
LD = lx


all:mre_api.elf mrd_api.elf

mre_api.elf: crt4100.o mre_api.o makeapi
	$(LD) -o mre_api.elf -e __start -sec { .text 0x80010000 : .MRETEXT : .data
0x84000000 : .MREDATA : .sdata : .sbss : .bss } crt4100.o mre_api.o -L../../lib4100 -lmhmr -M

mrd_api.elf: crt4100.o mrd_api.o makeapi
	$(LD) -o mrd_api.elf -e __start -sec { .text 0x80010000 : .MRDTEXT : .data
0x84000000 : .MRDDATA : .sdata : .sbss : .bss } crt4100.o mrd_api.o -L../../lib4100 -lmhmr -M


mre_api.o: mre_api.c makeapi
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mre_api.o mre_api.c

mrd_api.o: mrd_api.c makeapi
	$(CC) -elf -cpu=r4100 -c -OA -ansi -G -o mrd_api.o mrd_api.c

crt4100.o: crt4100.s makeapi
	$(AS) -elf -r4100 -b0 -o crt4100.o crt4100.s
```

（ａ）GHS製リンカのオプションについて

-o〈ファイル名〉

生成される実行ファイル名の指定を行います。

-sec ｛セクション　アドレス［：セクション　アドレス...］｝

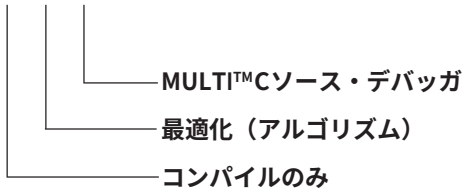
セクション（.text, .data, ...）の先頭アドレスを設定します。各セクションの指定は“：”で区切ります。

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について

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　ccmipel -c -OA -G mh_enc.c



詳細は，GHSのリンカ，コンパイラのマニュアルを参照してください。

（ｃ）make時の注意

makefileは次に示すように指定してください。

make -f makenorm（APIライブラリを使用しない場合）

make -f makeapi（APIライブラリを使用する場合）

３.２.２　V810ファミリ用サンプル

（１）NEC製リンカによるリンク

AP70732-B01のmakefileは次のようになっています。

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

all：mh_enc.elf mr_enc.elf mmr_enc.elf mh_dec.elf mr_dec.elf mmr_dec.elf
mh_enc.elf：crt810.o mh_enc.o
    $（LD）-D mr_enc.lnk crt810.o mh_enc.o libmhmr.a -o mh_enc.elf

mr_enc.elf：crt810.o mr_enc.o
    $（LD）-D mr_enc.lnk crt810.o mmr_enc.o libmhmr.a -o mr_enc.elf

mmr_enc.elf：crt810.o mmr_enc.o
    $（LD）-D mr_enc.lnk crt810.o mmr_enc.o libmhmr.a -o mmr_enc.elf

mh_dec.elf：crt810.o mh_dec.o
    $（LD）-D mr_dec.lnk crt810.o mh_dec.o libmhmr.a -o mh_dec.elf

mr_dec.elf：crt810.o mr_dec.o
    $（LD）-D mr_dec.lnk crt810.o mr_dec.o libmhmr.a -o mr_dec.elf

mmr_dec.elf：crt810.o mmr_dec.o
    $（LD）-D mr_dec.lnk crt810.o mmr_dec.o libmhmr.a -o mmr_dec.elf

mh_enc.o：mh_enc.c
    $（CC）-Wa, -cn -cpu 742 -c mh_enc.c

mr_enc.o：mr_enc.c
    $（CC）-Wa, -cn -cpu 742 -c mr_enc.c

mmr_enc.o：mmr_enc.c
    $（CC）-Wa, -cn -cpu 742 -c mmr_enc.c

mh_dec.o：mh_dec.c
    $（CC）-Wa, -cn -cpu 742 -c mh_dec.c

mr_dec.o：mr_dec.c
    $（CC）-Wa, -cn -cpu 742 -c mr_dec.c

mmr_dec.o：mmr_dec.c
    $（CC）-Wa, -cn -cpu 742 -c mmr_dec.c

crt810.c：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など）の先頭アドレスを設定します。

以下に，mr_enc.lnkの内容を示します。

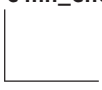
```
DATA：!LOAD ?RW V0x0     {.data = $PROGBITS ?AW；MREDATA = $PROGBITS ?AW;}；
TEXT：!LOAD ?RX V0x10000{.text = $PROGBITS ?AX；MRETEXT = $PROGBITS ?AX;}；
_ _tp_TEXT @ %TP_SYMBOL;
_ _gp_DATA @ %GP_SYMBOL &_ _tp_TEXT;
```

以下に，mr_dec.lnkの内容を示します。

```
DATA：!LOAD ?RW V0x0     {.data = $PROGBITS ?AW；MRDDATA = $PROGBITS ?AW;}；
TEXT：!LOAD ?RX V0x10000{.text = $PROGBITS ?AX；MRDTEXT = $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 mh_enc.c



コンパイルだけ

詳細は，**CA732　ユーザーズ・マニュアル　操作編**を参照してください。

（2）GHS製リンカによるリンク

AP70732-B01のmakefileは次のようになっています。

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

all：mh_enc.elf mr_enc.elf mmr_enc.elf mh_dec.elf mr_dec.elf mmr_dec.elf
mh_enc.elf：crt810.o reset.o mh_enc.o
$（LD）-o mh_enc.elf -e  _ _start -sec  {.text 0x10000：.MRETEXT：.data 0x0：.MREDATA：.sdata：
.sbss：.bss：reset 0xffffffec}  crt810.o mh_enc.o -L../../lib810 -lmhmr reset.o -M

mr_enc.elf：crt810.o reset.o mr_enc.o
$（LD）-o mr_enc.elf -e  _ _start -sec  {.text 0x10000：.MRETEXT：.data 0x0：.MREDATA：.sdata：
.sbss：.bss：reset 0xffffffec}  crt810.o mr_enc.o -L../../lib810 -lmhmr reset.o -M

mmr_enc.elf：crt810.o reset.o mmr_enc.o
$（LD）-o mmr_enc.elf -e  _ _start -sec  {.text 0x10000：.MRETEXT：.data 0x0：.MREDATA：.sdata：
.sbss：.bss：reset 0xffffffec}  crt810.o mmr_enc.o -L../../lib810 -lmhmr reset.o -M

mh_dec.elf：crt810.o reset.o mh_dec.o
$（LD）-o mh_dec.elf -e  _ _start -sec  {.text 0x10000：.MRDTEXT：.data 0x0：.MRDDATA：.sdata：
.sbss：.bss：reset 0xffffffec}  crt810.o mh_dec.o -L../../lib810 -lmhmr reset.o -M

mr_dec.elf：crt810.o reset.o mr_dec.o
$（LD）-o mr_dec.elf -e  _ _start -sec  {.text 0x10000：.MRDTEXT：.data 0x0：.MRDDATA：.sdata：
.sbss：.bss：reset 0xffffffec}  crt810.o mr_dec.o -L../../lib810 -lmhmr reset.o -M

mmr_dec.elf：crt810.o reset.o mmr_dec.o
$（LD）-o mmr_dec.elf -e  _ _start -sec  {.text 0x10000：.MRDTEXT：.data 0x0：.MRDDATA：.sdata：
.sbss：.bss：reset 0xffffffec}  crt810.o mmr_dec.o -L../../lib810 -lmhmr reset.o -M

mh_enc.o：mh_enc.c
$（CC）-c -OA -G mh_enc.c

mr_enc.o：mr_enc.c
$（CC）-c -OA -G mr_enc.c

mmr_enc.o：mmr_enc.c
$（CC）-c -OA -G mmr_enc.c

mh_dec.o：mh_dec.c
$（CC）-c -OA -G mh_dec.c

mr_dec.o：mr_dec.c
$（CC）-c -OA -G mr_dec.c

mmr_dec.o：mmr_dec.c
$（CC）-c -OA -G mmr_dec.c

crt810.o：crt810.s
$（AS）-elf -cpu=V810 -o crt810.o crt810.s

reset.o：reset.s
$（AS）-elf -cpu=V810 -o reset.o reset.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...］｝

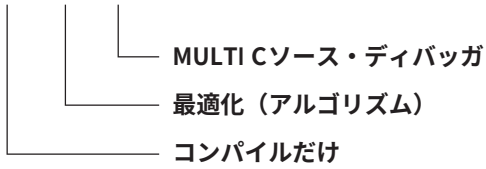
セクション（.text, .data, ...）の先頭アドレスを設定します。各セクションの指定は“：”で区切ります。

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，コンパイルを行います。

例　cc810　-c　-OA　-G　mh_enc.c



詳細は，GHSのリンカ，コンパイラのマニュアルを参照してください。

３.２.３　V830ファミリ用サンプル

（１）NEC製リンカによるリンク

AP705100-B01のmakefileは次のようになっています。

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

all：mh_enc.elf mh_dec.elf mr_enc.elf mr_dec.elf mmr_enc.elf mmr_dec.elf

mh_enc.elf：crt830.o mh_enc.o mr_enc.lnk
    $（LD）-D mr_enc.lnk -cpu 5100 crt830.o mh_enc.o libmhmr.a -o mh_enc.elf

mh_dec.elf：crt830.o mh_dec.o mr_dec. lnk
    $（LD）-D mr_dec.lnk -cpu 5100 crt830.o mh_dec.o libmhmr.a -o mh_dec.elf

mr_enc.elf：crt830.o mr_enc.o mr_enc.lnk
    $（LD）-D mr_enc.lnk -cpu 5100 crt830.o mr_enc.o libmhmr.a -o mr_enc.elf

mr_dec.elf：crt830.o mr_dec.o mr_dec. lnk
    $（LD）-D mr_dec.lnk -cpu 5100 crt830.o mr_dec.o libmhmr.a -o mr_dec.elf

mmr_enc.elf：crt830.o mmr_enc.o mr_enc.lnk
    $（LD）-D mr_enc.lnk -cpu 5100 crt830.o mmr_enc.o libmhmr.a -o mmr_enc.elf

mmr_dec.elf：crt830.o mmr_dec.o mr_dec. lnk
    $（LD）-D mr_dec.lnk -cpu 5100 crt830.o mmr_dec.o libmhmr.a -o mmr_dec.elf

mh_enc.o：mh_enc.c
    $（CC）-cpu 5100 -c mh_enc.c

mh_dec.o：mh_dec.c
    $（CC）-cpu 5100 -c mh_dec.c

mr_enc.o：mr_enc.c
    $（CC）-cpu 5100 -c mr_enc.c

mr_dec.o：mr_dec.c
    $（CC）-cpu 5100 -c mr_dec.c

mmr_enc.o：mmr_enc.c
    $（CC）-cpu 5100 -c mmr_enc.c

mmr_dec.o：mmr_dec.c
    $（CC）-cpu 5100 -c mmr_dec.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，mr_enc.lnkの内容を示します。

```
TEXT：!LOAD ?RX V0×10000000{.text = $PROGBITS ?AX; .MRETEXT = $PROGBITS ?AX;};
DATA2：!LOAD ?RW V0×10008000{.data = $PROGBITS ?AW;};
DATA1：!LOAD ?RW V0×10010000{.MREDATA = $PROGBITS ?AW;};
_ _tp_TEXT @ %TP_SYMBOL；
_ _gp_DATA @ %GP_SYMBOL &_ _tp_TEXT；
```

以下に，mr_dec.lnkの内容を示します。

```
TEXT：!LOAD ?RX V0×10000000{.text = $PROGBITS ?AX; .MRDTEXT = $PROGBITS ?AX;};
DATA2：!LOAD ?RW V0×10008000{.data = $PROGBITS ?AW;};
DATA1：!LOAD ?RW V0×10010000{.MRDDATA = $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　-c　mh_enc.c

-c ―― コンパイルだけ

詳細は，**CA830　ユーザーズ・マニュアル　操作編**を参照してください。

（2）GHS製リンカによるリンク

AP705100-B01のmakefileは次のようになっています。

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

all：mh_enc.elf mh_dec.elf mr_enc.elf mr_dec.elf mmr_enc.elf mmr_dec.elf

mh_enc.elf：crt830.o reset.o mh_enc.o
$（LD）-o mh_enc.elf -e  _ _start -sec  {.text 0x10000000：.MRETEXT：.data 0x10008000：.MREDATA
0x10010000：.macinfo：.line：.debug：.sdata：.sbss：.bss：reset 0xfffffff0}  crt830.o mh_enc.o -L../../
lib830 -lmhmr reset.o -M

mh_dec.elf：crt830.o reset.o mh_dec.o
$（LD）-o mh_dec.elf -e  _ _start -sec  {.text 0x10000000：.MRDTEXT：.data 0x10008000：.MRDDATA
0x10010000：.macinfo：.line：.debug：.sdata：.sbss：.bss：reset 0xfffffff0}  crt830.o mh_dec.o -L../../
lib830 -lmhmr reset.o -M

mr_enc.elf：crt830.o reset.o mr_enc.o
$（LD）-o mr_enc.elf -e  _ _start -sec  {.text 0x10000000：.MRETEXT：.data 0x10008000：.MREDATA
0x10010000：.macinfo：.line：.debug：.sdata：.sbss：.bss：reset 0xfffffff0}  crt830.o mr_enc.o -L../../
lib830 -lmhmr reset.o -M

mr_dec.elf：crt830.o reset.o mr_dec.o
$（LD）-o mr_dec.elf -e  _ _start -sec  {.text 0x10000000：.MRDTEXT：.data 0x10008000：.MRDDATA
0x10010000：.macinfo：.line：.debug：.sdata：.sbss：.bss：reset 0xfffffff0}  crt830.o mr_dec.o -L../../
lib830 -lmhmr reset.o -M

mmr_enc.elf：crt830.o reset.o mmr_enc.o
$（LD）-o mmr_enc.elf -e  _ _start -sec  {.text 0x10000000：.MRETEXT：.data 0x10008000：.MREDATA
0x10010000：.macinfo：.line：.debug：.sdata：.sbss：.bss：reset 0xfffffff0}  crt830.o mmr_enc.o -L../../
lib830 -lmhmr reset.o -M

mmr_dec.elf：crt830.o reset.o mmr_dec.o
$（LD）-o mmr_dec.elf -e  _ _start -sec  {.text 0x10000000：.MRDTEXT：.data 0x10008000：.MRDDATA
0x10010000：.macinfo：.line：.debug：.sdata：.sbss：.bss：reset 0xfffffff0}  crt830.o mmr_dec.o -L../../
lib830 -lmhmr reset.o -M

mh_enc.o：mh_enc.c
$（CC）-c -OA -G mh_enc.o mh_enc.c

mh_dec.o：mh_dec.c
$（CC）-c -OA -G mh_dec.o mh_dec.c

mr_enc.o：mr_enc.c
$（CC）-c -OA -G mr_enc.o mr_enc.c

mr_dec.o：mr_dec.c
$（CC）-c -OA -G mr_dec.o mr_dec.c

mmr_enc.o：mmr_enc.c
$（CC）-c -OA -G mmr_enc.o mmr_enc.c

mmr_dec.o：mmr_dec.c
$（CC）-c -OA -G mmr_dec.o mmr_dec.c

crt830.o：crt830.s
$（AS）-w -elf -cpu=V830 -o crt830.o crt830.s

reset.o：reset.s
$（AS）-elf -cpu=V830 -o reset.o reset.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...］｝

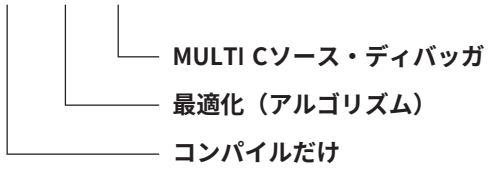
セクション（.text, .data, ...）の先頭アドレスを設定します。各セクションの指定は“：”で区切ります。

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，コンパイルを行います。

例　cc830　-c　-OA　-G　enc_main.c



詳細は，**GHSリンカ，コンパイラのマニュアル**を参照してください。

３.２.４　V850ファミリ用サンプル

（１）NEC製リンカによるリンク

AP703000-B01のmakefileは次のようになっています。

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

all：mh_enc.elf mr_enc.elf mmr_enc.elf mh_dec.elf mr_dec.elf mmr_dec.elf
mh_enc.elf：crt850.o mh_enc.o
    $（LD）-D mr_enc.lnk crt850.o mh_enc.o libmhmr.a -o mh_enc.elf

mr_enc.elf：crt850.o mr_enc.o
    $（LD）-D mr_enc.lnk crt850.o mr_enc.o libmhmr.a -o mr_enc.elf

mmr_enc.elf：crt850.o mmr_enc.o
    $（LD）-D mr_enc.lnk crt850.o mmr_enc.o libmhmr.a -o mmr_enc.elf

mh_dec.elf：crt850.o mh_dec.o
    $（LD）-D mr_dec.lnk crt850.o mh_dec.o libmhmr.a -o mh_dec.elf

mr_dec.elf：crt850.o mr_dec.o
    $（LD）-D mr_dec.lnk crt850.o mr_dec.o libmhmr.a -o mr_dec.elf

mmr_dec.elf：crt850.o mmr_dec.o
    $（LD）-D mr_dec.lnk crt850.o mmr_dec.o libmhmr.a -o mmr_dec.elf

mh_enc.o：mh_enc.c
    $（CC）-cpu 3000 -c mh_enc.c

mr_enc.o：mr_enc.c
    $（CC）-cpu 3000 -c mr_enc.c

mmr_enc.o：mmr_enc.c
    $（CC）-cpu 3000 -c mmr_enc.c

mh_dec.o：mh_dec.c
    $（CC）-cpu 3000 -c mh_dec.c

mr_dec.o：mr_dec.c
    $（CC）-cpu 3000 -c mr_dec.c

mmr_dec.o：mmr_dec.c
    $（CC）-cpu 3000 -c mmr_dec.c

crt850.o：crt850.s
    $（AS）-cn -cpu 3000 crt850.s -o crt850.o
```

保守／廃止

（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，mr_enc.lnkの内容を示します。

```
TEXT：!LOAD ?RX V0x2000{.text = $PROGBITS ?AX;.MRETEXT = $PROGBITS ?AX;};
DATA1：!LOAD ?RW V0x100000{.MREDATA = $PROGBITS ?AW;};
DATA2：!LOAD ?RW V0x110000{.data = $PROGBITS ?AW;};
_ _tp_TEXT @ %TP_SYMBOL；
_ _gp_DATA @ %GP_SYMBOL &_ _tp_TEXT；
_ _ep_DATA @ %EP_SYMBOL；
```

以下に，mr_dec.lnkの内容を示します。

```
TEXT：!LOAD ?RX V0x2000{.text = $PROGBITS ?AX;.MRDTEXT = $PROGBITS ?AX;};
DATA1：!LOAD ?RW V0x100000{.MRDDATA = $PROGBITS ?AW;};
DATA2：!LOAD ?RW V0x110000{.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 mh_enc.c

コンパイルだけ

詳細は**CA850　ユーザーズ・マニュアル　操作編**を参照してください。

（2）GHS製リンカによるリンク

AP703000-B01のmakefileは次のようになっています。

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

all：mh_enc.elf mr_enc.elf mmr_enc.elf mh_dec.elf mr_dec.elf mmr_dec.elf
mh_enc.elf：crt850.o reset.o mh_enc.o
$（LD）-o mh_enc.elf -e _ _start -sec {.text 0x2000：.MRETEXT：.MREDATA 0x100000：.data
0x110000：.sdata：.sbss：.bss：.reset 0x0} crt850.o mh_enc.o -L../../lib850 -lmhmr reset.o -M

mr_enc.elf：crt850.o reset.o mr_enc.o
$（LD）-o mr_enc.elf -e _ _start -sec {.text 0x2000：.MRETEXT：.MREDATA 0x100000：.data
0x110000：.sdata：.sbss：.bss：.reset 0x0} crt850.o mr_enc.o -L../../lib850 -lmhmr reset.o -M

mmr_enc.elf：crt850.o reset.o mmr_enc.o
$（LD）-o mmr_enc.elf -e _ _start -sec {.text 0x2000：.MRETEXT：.MREDATA 0x100000：.data
0x110000：.sdata：.sbss：.bss：.reset 0x0} crt850.o mmr_enc.o -L../../lib850 -lmhmr reset.o -M

mh_dec.elf：crt850.o reset.o mh_dec.o
$（LD）-o mh_dec.elf -e _ _start -sec {.text 0x2000：.MRDTEXT：.MRDDATA 0x100000：.data
0x110000：.sdata：.sbss：.bss：.reset 0x0} crt850.o mh_dec.o -L../../lib850 -lmhmr reset.o -M

mr_dec.elf：crt850.o reset.o mr_dec.o
$（LD）-o mr_dec.elf -e _ _start -sec {.text 0x2000：.MRDTEXT：.MRDDATA 0x100000：.data
0x110000：.sdata：.sbss：.bss：.reset 0x0} crt850.o mr_dec.o -L../../lib850 -lmhmr reset.o -M

mmr_dec.elf：crt850.o reset.o mmr_dec.o
$（LD）-o mmr_dec.elf -e _ _start -sec {.text 0x2000：.MRDTEXT：.MRDDATA 0x100000：.data
0x110000：.sdata：.sbss：.bss：.reset 0x0} crt850.o mmr_dec.o -L../../lib850 -lmhmr reset.o -M

mh_enc.o：mh_enc.c
$（CC）-c -OA -G mh_enc.c

mr_enc.o：mr_enc.c
$（CC）-c -OA -G mr_enc.c

mmr_enc.o：mmr_enc.c
$（CC）-c -OA -G mmr_enc.c

mh_dec.o：mh_dec.c
$（CC）-c -OA -G mh_dec.c

mr_dec.o：mr_dec.c
$（CC）-c -OA -G mr_dec.c

mmr_dec.o：mmr_dec.c
$（CC）-c -OA -G mmr_dec.c

crt850.o：crt850.s
$（AS）-elf -o crt850.o crt850.s

reset.o：reset.s
$（AS）-elf -o reset.o reset.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...］｝

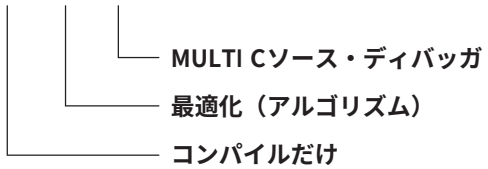
セクション（.text, .data, ...）の先頭アドレスを設定します。各セクションの指定は“：”で区切ります。

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，コンパイラのマニュアルを参照してください。

〔メ　モ〕

第４章　システム例

MH/MR/MMRの各符号化方式におけるシステム例を図４－１から図４－８に示します。また，メイン・ソースについては**付録A　MH/MR/MMRサンプル・ソース・リスト（AP30100-B01），付録B　MH/MR/MMRサンプル・ソース・リスト（AP703000-B01, AP705100-B01, AP70732-B01）**を参照してください。

４.１　圧 縮 系

４.１.１　MH方式

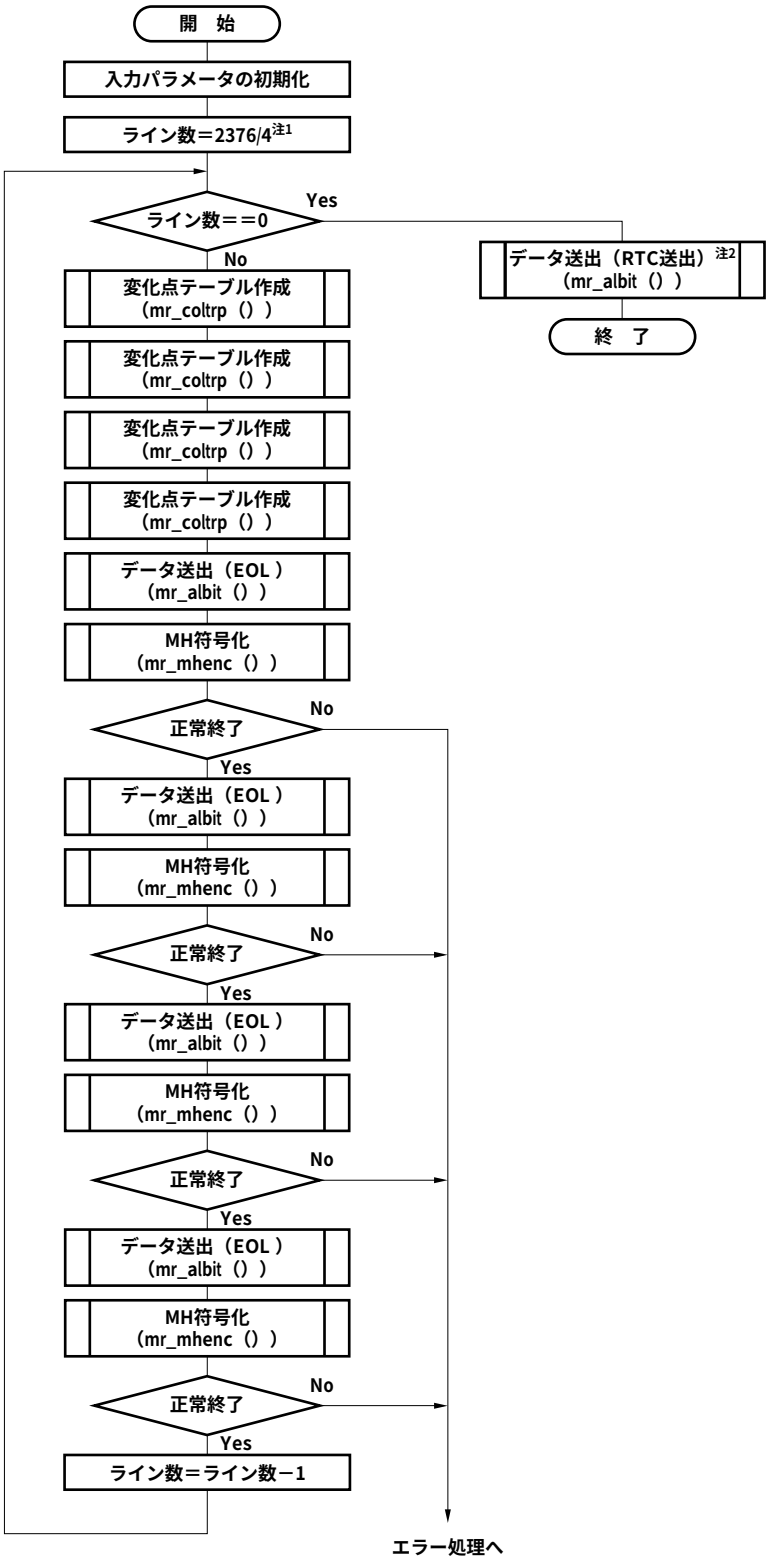
１ライン分の画素データを変化点テーブル作成処理により変化点情報に変換し，変化点テーブルに格納します。

次にデータ送出処理により送信バッファにEOLを送出し，１ライン分の変化点情報をMH符号化処理により符号化用テーブルを用いて符号データに変換し，送信バッファに送出します。なお，データ送出，MH符号化時に中断した場合は中断処理を行ってください。

処理フロー例を図４－１に示します。図４－１で変化点テーブルを４ライン連続して作成しているのは，変化点テーブル・アドレスの再設定処理を少なくするためです。MR方式についても同じです。

保守／廃止

図４－１　MH符号化の処理フロー



注１． 2376は，200 dpi，A4の場合における１ページ分のライン数を意味しています。

２． RTC送出はEOLを６回連続送出します。

保守／廃止

４．１．２　MR方式

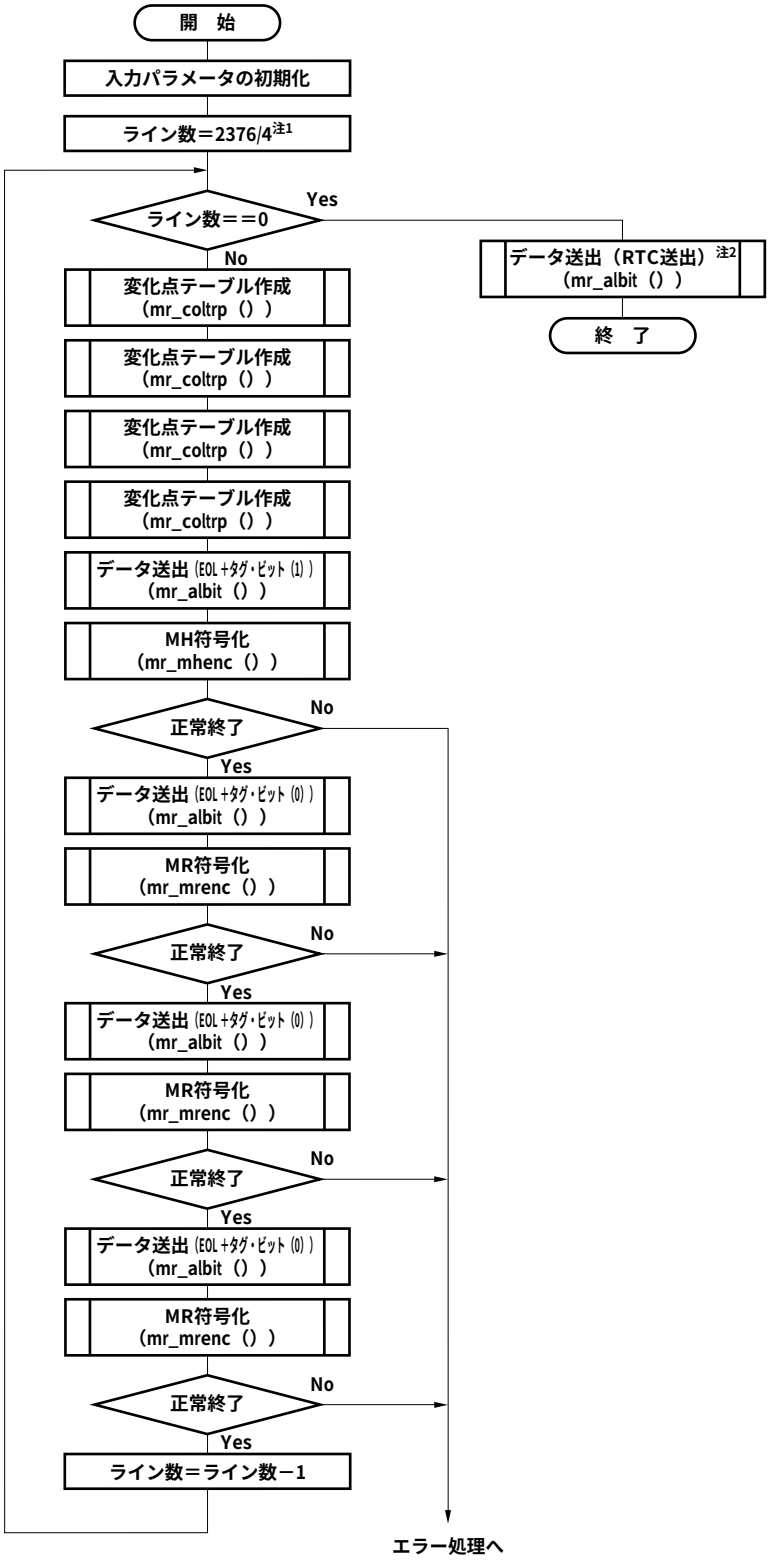
１ライン分の画素データを変化点テーブル作成処理により変化点情報に変換し，変化点テーブルに格納します。

次にデータ送出処理により送信バッファにEOL＋タグ・ビットを送出し，１ライン分の変化点情報をMH/MR符号化処理により符号化用テーブルを用いて符号データに変換し，送信バッファに送出します。なお，データ送出，MH/MR符号化時に中断した場合は中断処理を行ってください。

処理フロー例を図４－２に示します。

保守／廃止

図４－２　MR符号化の処理フロー



注１．2376は，200 dpi，A4の場合における１ページ分のライン数を意味しています。

２．RTC送出はEOL＋タグ・ビット（１）を６回連続送出します。

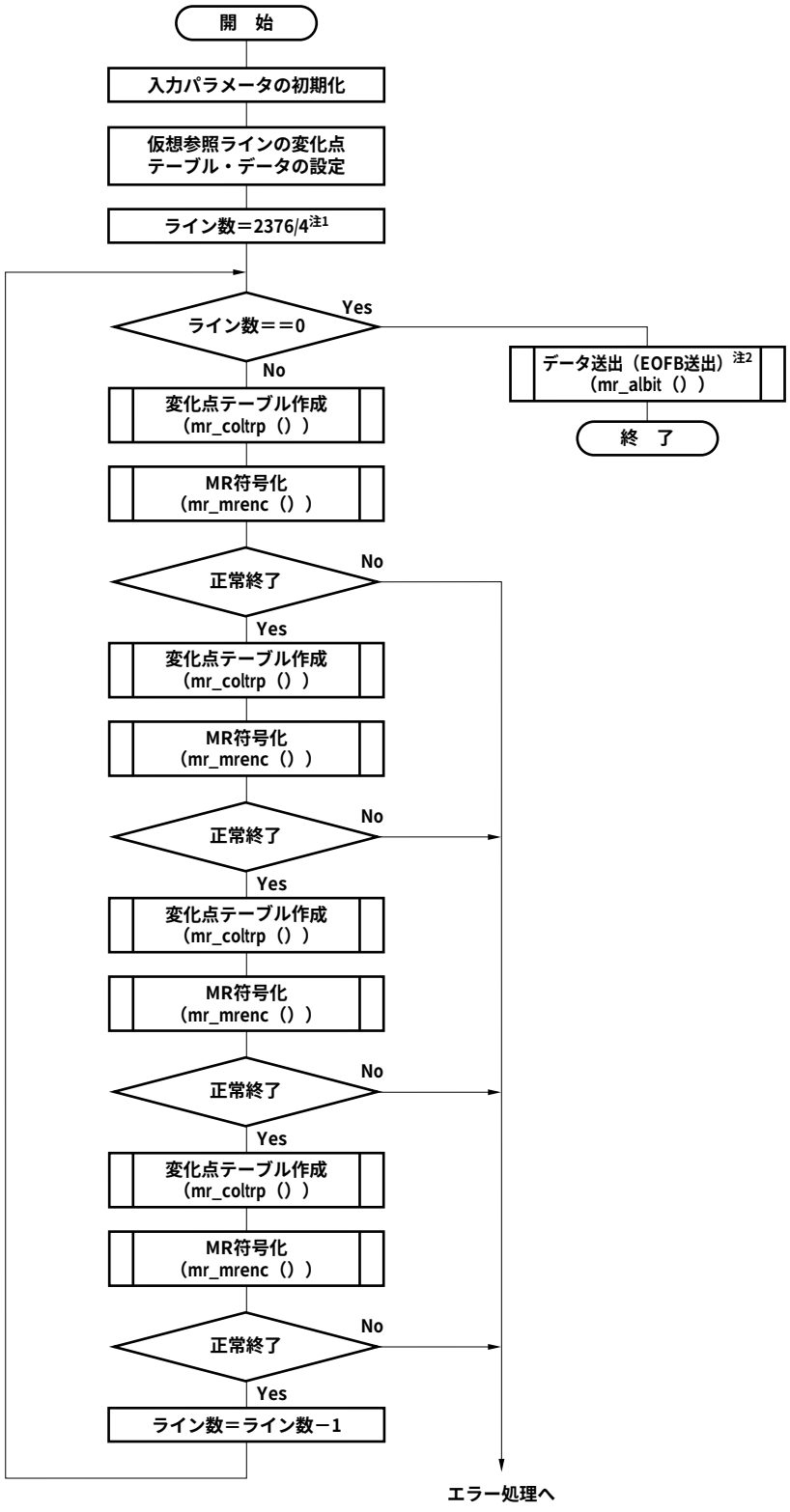
４.１.３　MMR方式

１ライン分の画素データを変化点テーブル作成処理により変化点情報に変換し，変化点テーブルに格納します。

次に１ライン分の変化点情報をMR符号化処理により，符号化用テーブルを用いて符号データに変換し，送信バッファに送出します。なお，データ送出，MR符号化時に中断した場合は中断処理を行ってください。

処理フロー例を図４－３に示します。

図４－３　MMR符号化の処理フロー



注１．2376は，200 dpi，A4の場合における１ページ分のライン数を意味しています。

２．EOFB送出はEOLを２回連続送出します。

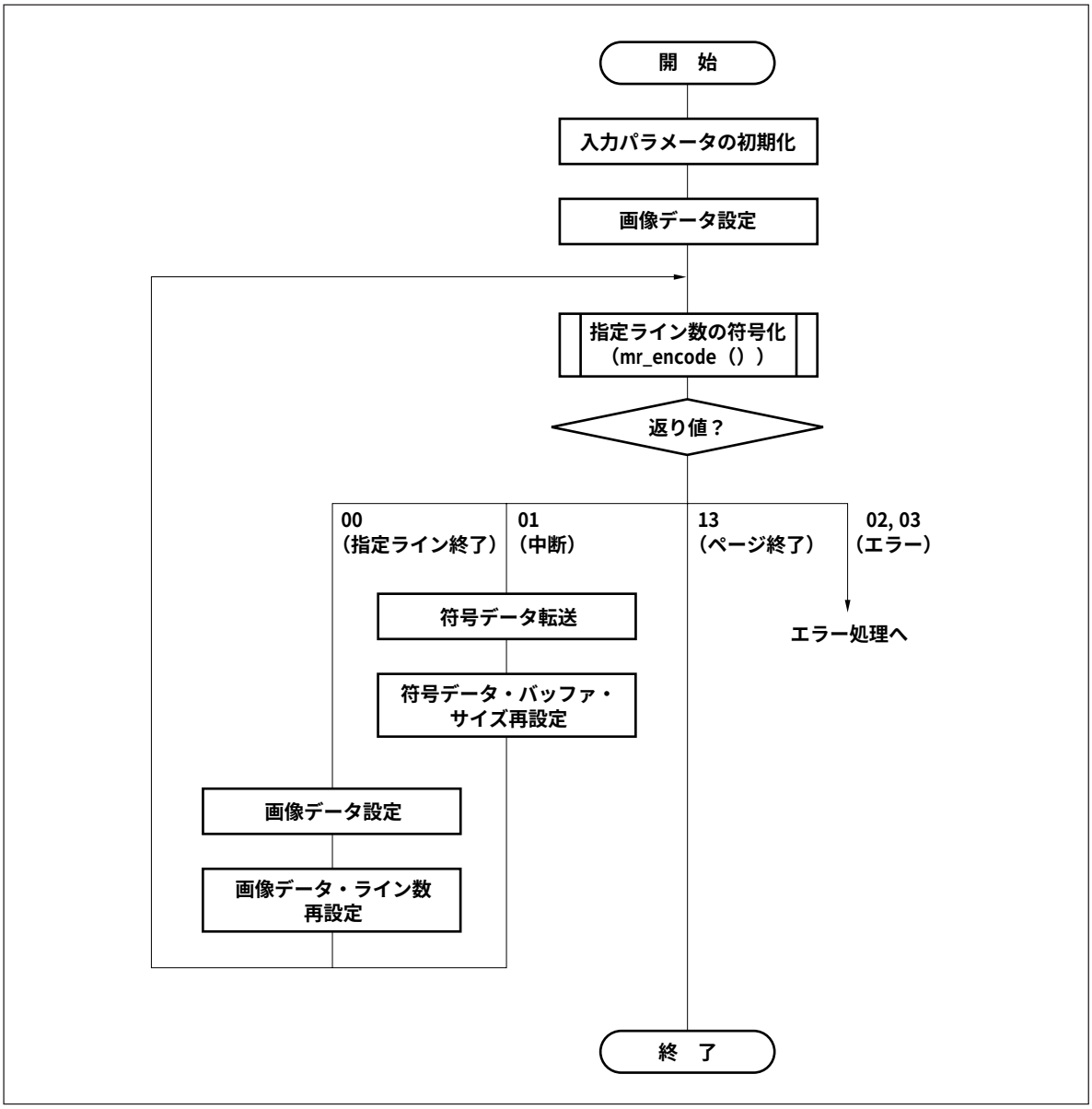
保守／廃止

★　４．１．４　APIライブラリを使用したMR方式（AP30100-B01のみ）

指定ラインごとに画像データを設定して符号化処理を行い，送信バッファに送出します。符号データ・バッファ・サイズが０になると中断します。符号データを転送し，符号データ・バッファ・サイズを再設定して再開します。指定ライン分の符号化が終了した場合は，次の画像データを設定し，画像データ・ライン数を再設定して再開します。１ページ分の処理が終了するとRTC符号を送出して，１ページの処理を終了します。そのほかの返り値で中断した場合は中断処理を行ってください。

処理フロー例を図４－４に示します。

図４－４　APIライブラリを使用したMR符号化の処理フロー



保守／廃止

4.2　伸 長 系

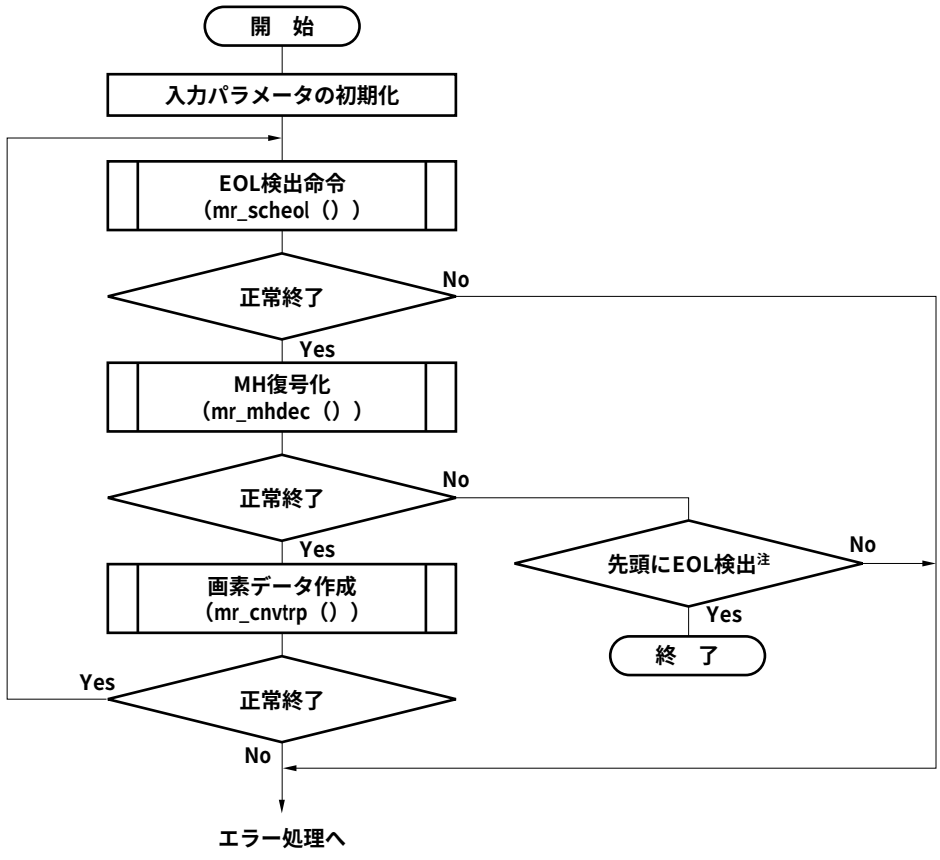
4.2.1　MH方式

EOL検出処理を用いて符号データからEOLを検出します。

次に符号データをMH復号化処理により，１ライン分の変化点情報に変換し，変化点テーブルに格納します。このとき，符号化データは復号化用テーブルを用いて変化点情報に変換します。この変化点情報を画素データ作成処理により，画素データに変換し，プリント・バッファに送出します。なお，EOL検出処理，MH復号化時に中断した場合は中断処理を行ってください。

処理フロー例を図４－５に示します。

図４－５　MH復号化の処理フロー



注　RTCは６個のEOLで検出します。

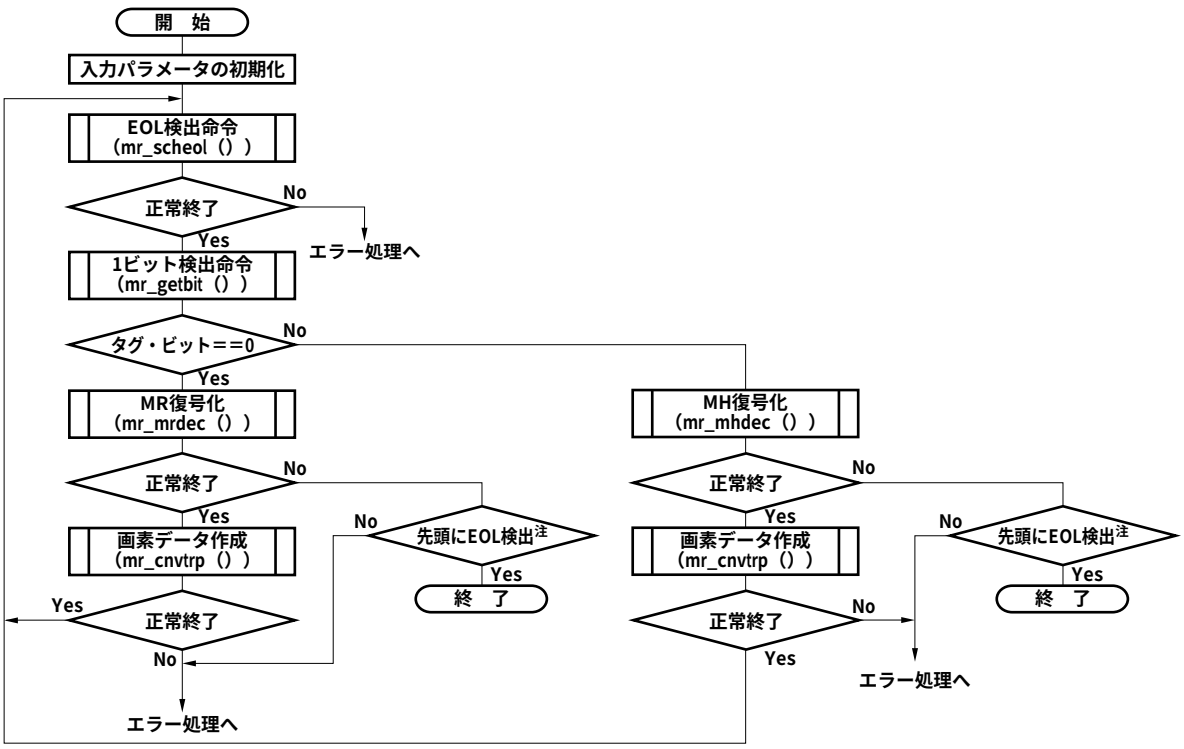
保守／廃止

４.２.２　MR方式

EOL検出処理を用いて符号データからEOLを検出します。次に１ビット検出処理によりタグ・ビットを検出します。タグ・ビットが“１”の場合はMH復号化処理を行い，“０”の場合はMR復号化処理を行い，１ライン分の変化点情報に変換し，変化点テーブルに格納します。このとき，符号化データは復号化用テーブルを用いて変化点情報に変換します。この変化点情報を画素データ作成処理により画素データに変換し，プリント・バッファに送出します。なお，EOL検出処理，MH/MR復号化時に中断した場合は中断処理を行ってください。

処理フロー例を図４－６に示します。

図４－６　MR復号化の処理フロー



注　RTCは６個のEOL＋タグ・ビット（１）で検出します。

保守／廃止

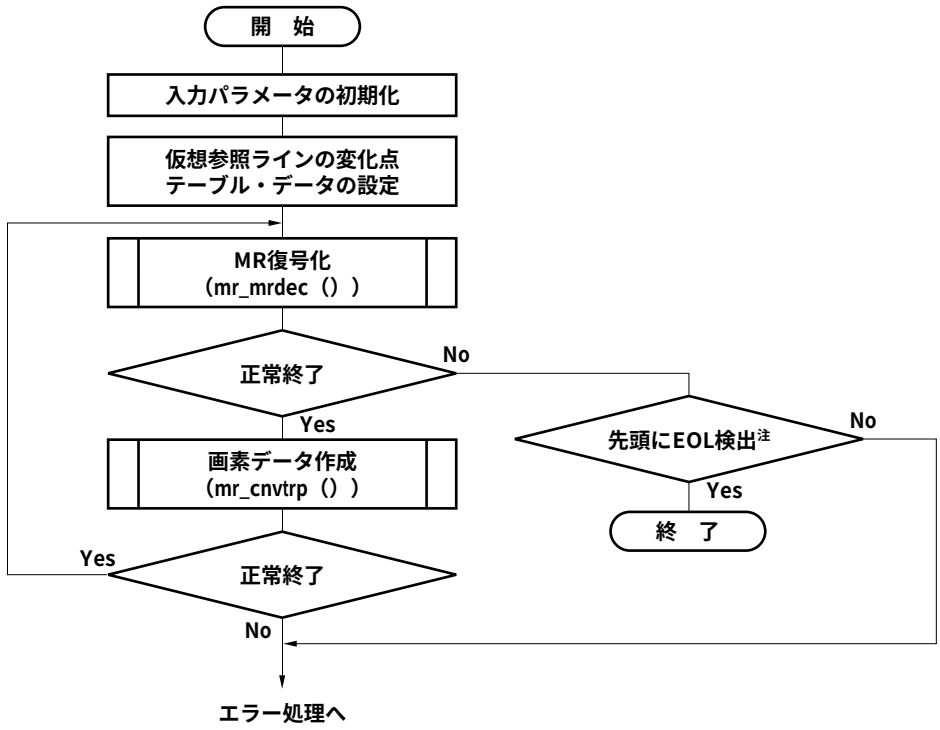
４.２.３　MMR方式

仮想ラインの変化点情報（１ラインすべて白）を変化点テーブルに書き込みます。

次に符号データをMR復号化処理により，１ライン分の変化点情報に変換し変化点テーブルに格納します。このとき，符号化データは復号化用テーブルを用いて変化点情報に変換します。この変化点情報を画素データ作成処理により，画素データに変換し，プリント・バッファに送出します。なお，EOL検出処理，MR復号化時に中断した場合は中断処理を行ってください。

処理フロー例を図４－７に示します。

図４－７　MMR復号化の処理フロー



注　EOFBは２個のEOLで検出します。

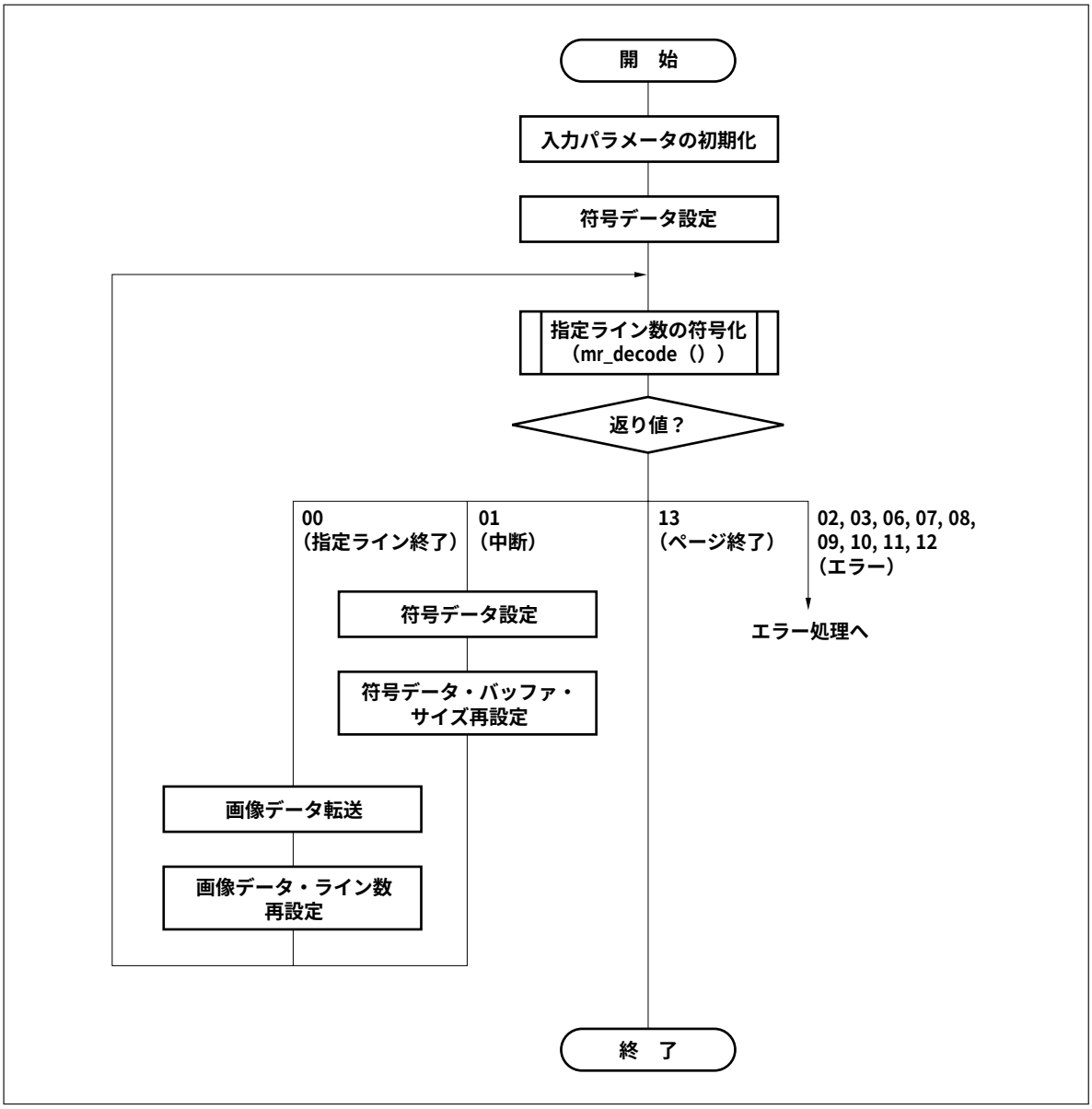
保守／廃止

★　４．２．４　APIライブラリを使用したMR方式（AP30100-B01のみ）

設定した符号データから指定ライン分の復号化処理を行い，プリント・バッファに送出します。符号データ・バッファ・サイズが０になると中断します。次の符号データを設定し，符号データ・バッファ・サイズを再設定して再開します。指定ライン分の復号化が終了した場合は，画像データを転送し，画像データ・ライン数を再設定して再開します。RTC符号を検出すると１ページ分の処理が終了します。その他の返り値で中断した場合は中断処理を行ってください。

処理フロー例を図４－８に示します。

図４－８　APIライブラリを使用したMR符号化の処理フロー



保守／廃止

４.３　メモリ・マップ例

図４－９から図４－12に，サンプル・プログラム（圧縮／伸長）のメモリ・マップを示します（図４－９から図４－12のメモリ・マップはGHS製リンカ）。

★　**図４－９　サンプル・プログラム（圧縮／伸長）のメモリ・マップ（AP30100-B01）：VR4100シリーズの場合**

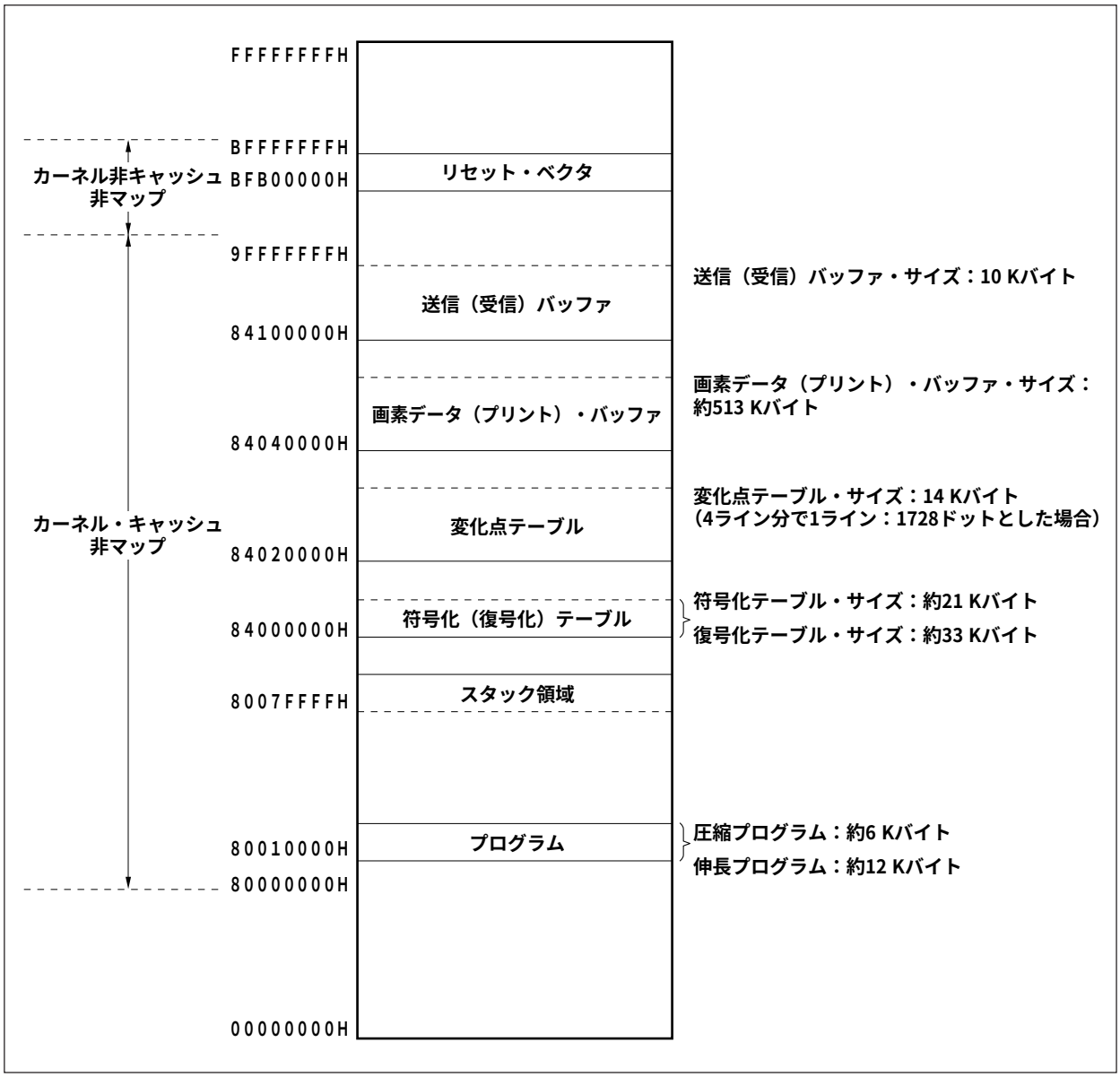
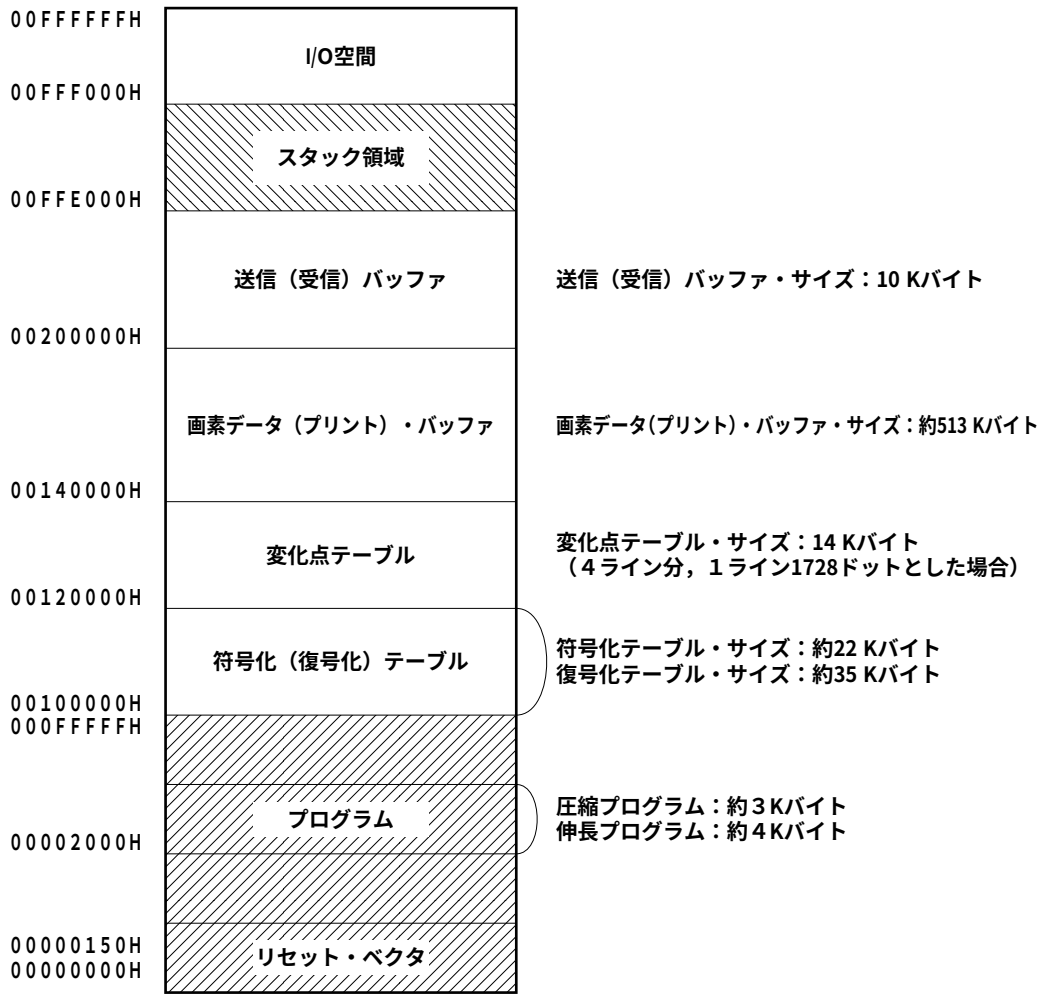


図４－10　サンプル・プログラム（圧縮／伸長）のメモリ・マップ（AP703000-B01）：V851の場合

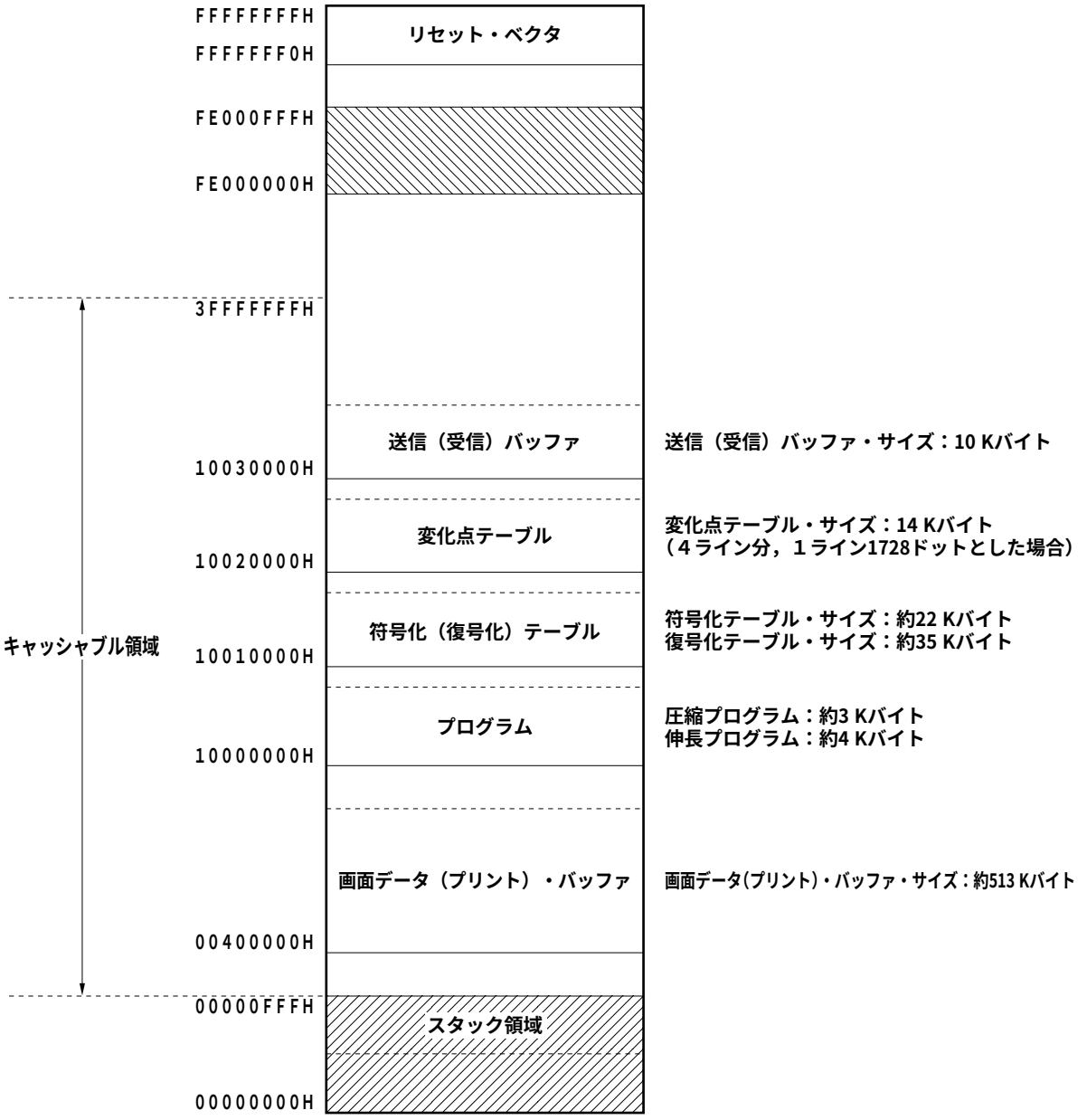


備考　　は，内蔵ROM空間（00000000H～000FFFFFH）です。

は，内蔵RAM空間（00FFE000H～00FFEFFFH）です。

保守／廃止

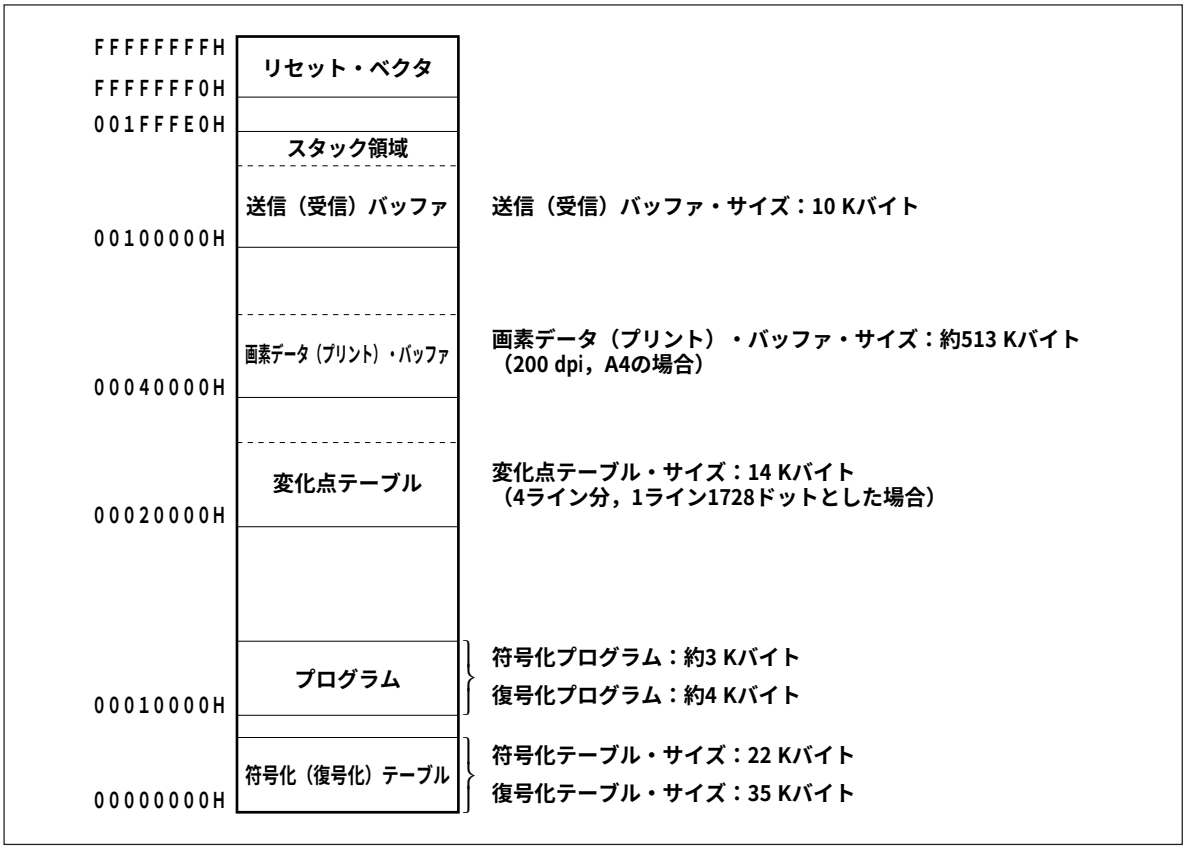
図４－11　サンプル・プログラム（圧縮／伸長）のメモリ・マップ（AP705100-B01）：V830の場合



備考　　は，内蔵データRAM空間（00000000H～00000FFFH）です。

は，内蔵命令RAM空間（FE000000H～FE000FFFH）です。

★　**図４－12　サンプル・プログラム（圧縮／伸長）のメモリ・マップ（AP70732-B01）：V810ファミリの場合**



4.4　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“mr_”が付けられています。“mr_”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〕

★ 付録A　MH/MR/MMRサンプル・ソース・リスト（AP30100-B01）

（1）MH方式圧縮　mh_enc.c

```
#include "codec_vr.h"

#define BUF_SIZE            0x10000
#define ENC_BUF             0x84100000L
#define PIXEL_BUF           0x84040000L
#define PIXEL_NUM           1728
#define RUN_TBL             0x84020000L
#define RUN_TBL_BUF_SIZE    0x20000L
#define LINE_CNT            2376
#define RTC_CNT             6

void main()
{
    MREINFO mreinfo;
    register unsigned int cnt;
    unsigned int err_code = 0;

    mreinfo.odd_data = 0;
    mreinfo.odd_bit = 0;
    mreinfo.enc_buf = (unsigned int *)ENC_BUF;
    mreinfo.pixel_buf = (unsigned int *)PIXEL_BUF;
    mreinfo.enc_buf_size = BUF_SIZE;
    mreinfo.pixel_num = PIXEL_NUM;
    mreinfo.restart_adr = 0;
    mreinfo.send_data = 0x800;
    mreinfo.send_bit = 12;

    for( cnt = (LINE_CNT/4) ; cnt != 0 ; cnt-- ){
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            mreinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
            if( mr_coltrp( &mreinfo ))buf_size0_err();
            if( mr_coltrp( &mreinfo ))buf_size0_err();
            if( mr_coltrp( &mreinfo ))buf_size0_err();
            if( mr_coltrp( &mreinfo ))buf_size0_err();
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            mreinfo.restart_adr = 0;
            if( mr_albit( &mreinfo ))mreinfo.enc_buf_size = BUF_SIZE;
            while( err_code = mr_mhenc( &mreinfo )){
                switch(err_code){
                    case 1:mreinfo.enc_buf_size=BUF_SIZE;continue;
                    case 2:pix_min_err();break;
                    case 3:pix_over_err();break;
                }
            }
            if( mr_albit( &mreinfo ))mreinfo.enc_buf_size = BUF_SIZE;
```

```
            while( err_code = mr_mhenc( &mreinfo )){
                switch(err_code){
                    case 1:mreinfo.enc_buf_size = BUF_SIZE;continue;
                    case 2:pix_min_err();break;
                    case 3:pix_over_err();break;
                }
            }
            if( mr_albit( &mreinfo ))mreinfo.enc_buf_size = BUF_SIZE;
            while( err_code = mr_mhenc( &mreinfo )){
                switch(err_code){
                    case 1:mreinfo.enc_buf_size = BUF_SIZE;continue;
                    case 2:pix_min_err();break;
                    case 3:pix_over_err();break;
                }
            }
            if( mr_albit( &mreinfo ))mreinfo.enc_buf_size = BUF_SIZE;
            while( err_code = mr_mhenc( &mreinfo )){
                switch(err_code){
                    case 1:mreinfo.enc_buf_size = BUF_SIZE;continue;
                    case 2:pix_min_err();break;
                    case 3:pix_over_err();break;
                }
            }
    }
    for( cnt = RTC_CNT ; cnt != 0 ; cnt-- ){
        if( mr_albit( &mreinfo ))mreinfo.enc_buf_size = BUF_SIZE;
    }
    if( mreinfo.odd_bit)  *(mreinfo.enc_buf) = mreinfo.odd_data;

}


pix_min_err(){
    exit(1);
}

pix_over_err(){
    exit(1);
}

buf_size0_err(){
    exit(1);
}
```

（2）MR方式圧縮　mr_enc.c

```
#include "codec_vr.h"

#define BUF_SIZE            0x10000
#define ENC_BUF             0x84100000L
#define PIXEL_BUF           0x84040000L
#define PIXEL_NUM           1728
#define RUN_TBL             0x84020000L
#define RUN_TBL_BUF_SIZE    0x20000L
#define LINE_CNT            2376
#define RTC_CNT             6

void main()
{
    MREINFO mreinfo;
    register unsigned int cnt;
    unsigned int err_code = 0;

    mreinfo.odd_data = 0;
    mreinfo.odd_bit = 0;
    mreinfo.enc_buf = (unsigned int *)ENC_BUF;
    mreinfo.pixel_buf = (unsigned int *)PIXEL_BUF;
    mreinfo.enc_buf_size = BUF_SIZE;
    mreinfo.pixel_num = PIXEL_NUM;
    mreinfo.restart_adr = 0;

    mreinfo.send_bit = 13;

    for( cnt = (LINE_CNT/4) ; cnt != 0 ; cnt-- ){
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            mreinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
            if( mr_coltrp( &mreinfo )) buf_size0_err();
            if( mr_coltrp( &mreinfo )) buf_size0_err();
            if( mr_coltrp( &mreinfo )) buf_size0_err();
            if( mr_coltrp( &mreinfo )) buf_size0_err();
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            mreinfo.ref_tbl = (unsigned short *)RUN_TBL;
            mreinfo.send_data = 0x1800;
            if( mr_albit( &mreinfo )) mreinfo.enc_buf_size = BUF_SIZE;
            mreinfo.restart_adr = 0;
            while( err_code = mr_mhenc( &mreinfo )){
                switch(err_code){
                    case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
                    case 2: pix_min_err(); break;
                    case 3: pix_over_err(); break;
                }
            }
            mreinfo.send_data = 0x800;
            if( mr_albit( &mreinfo ))mreinfo.enc_buf_size = BUF_SIZE;
            while( err_code = mr_mrenc( &mreinfo )){
                switch(err_code){
                    case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
```

```
                case 2: pix_min_err(); break;
                case 3: pix_over_err(); break;
            }
        }
        if( mr_albit( &mreinfo )) mreinfo.enc_buf_size = BUF_SIZE;
        while( err_code = mr_mrenc( &mreinfo )){
            switch(err_code){
                case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
                case 2: pix_min_err(); break;
                case 3: pix_over_err(); break;
            }
        }
        if( mr_albit( &mreinfo )) mreinfo.enc_buf_size = BUF_SIZE;




        while( err_code = mr_mrenc( &mreinfo )){
            switch(err_code){
                case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
                case 2: pix_min_err(); break;
                case 3: pix_over_err(); break;
            }
        }
    }
    mreinfo.send_data = 0x1800;
    for( cnt = RTC_CNT ; cnt != 0 ; cnt-- ){
        if( mr_albit( &mreinfo )) mreinfo.enc_buf_size = BUF_SIZE;
    }
    if( mreinfo.odd_bit)  *(mreinfo.enc_buf) = mreinfo.odd_data;

}

pix_min_err(){
    exit(1);
}

pix_over_err(){
    exit(1);
}

buf_size0_err(){
    exit(1);
}
```

（3）MMR方式圧縮　mmr_enc.c

```
#include "codec_vr.h"

#define BUF_SIZE            0x10000
#define ENC_BUF             0x84100000L
#define PIXEL_BUF           0x84040000L
#define PIXEL_NUM           1728
#define RUN_TBL             0x84020000L
#define RUN_TBL_BUF_SIZE    0x84020000L
#define REF_TBL             0x84028000L
#define LINE_CNT            2376
#define RTC_CNT             2

void main()
{
    MREINFO mreinfo;
    register unsigned int cnt;
    unsigned int err_code = 0;

    mreinfo.odd_data = 0;
    mreinfo.odd_bit = 0;
    mreinfo.enc_buf = (unsigned int *)ENC_BUF;
    mreinfo.pixel_buf = (unsigned int *)PIXEL_BUF;
    mreinfo.enc_buf_size = BUF_SIZE;
    mreinfo.pixel_num = PIXEL_NUM;
    mreinfo.send_data = 0x800;
    mreinfo.send_bit = 12;
    mreinfo.run_tbl = (unsigned short *)RUN_TBL;
    mreinfo.ref_tbl = (unsigned short *)REF_TBL;
    *(mreinfo.ref_tbl) = PIXEL_NUM;
    *(mreinfo.ref_tbl + 1) = 0xffff;
    mreinfo.restart_adr = 0;

    for( cnt = (LINE_CNT/4) ; cnt != 0 ; cnt-- ){
            mreinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            mreinfo.ref_tbl = (unsigned short *)REF_TBL;

            if( mr_coltrp( &mreinfo ) ) buf_size0_err();
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            while( err_code = mr_mrenc( &mreinfo ) ){
                switch(err_code){
                    case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
                    case 2: pix_min_err(); break;
                    case 3: pix_over_err(); break;
                }
            }
            mreinfo.run_tbl = (unsigned short *)REF_TBL;
            mreinfo.ref_tbl = (unsigned short *)RUN_TBL;

            if( mr_coltrp( &mreinfo ) ) buf_size0_err();
```

```
            mreinfo.run_tbl = (unsigned short *)REF_TBL;
            while( err_code = mr_mrenc( &mreinfo ) ){
                switch(err_code){
                    case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
                    case 2: pix_min_err(); break;
                    case 3: pix_over_err(); break;
                }
            }
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            mreinfo.ref_tbl = (unsigned short *)REF_TBL;

            if( mr_coltrp( &mreinfo ) ) buf_size0_err();
            mreinfo.run_tbl = (unsigned short *)RUN_TBL;
            while( err_code = mr_mrenc( &mreinfo ) ){
                switch(err_code){
                    case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
                    case 2: pix_min_err(); break;
                    case 3: pix_over_err(); break;
                }
            }
            mreinfo.run_tbl = (unsigned short *)REF_TBL;
            mreinfo.ref_tbl = (unsigned short *)RUN_TBL;

            if( mr_coltrp( &mreinfo ) ) buf_size0_err();
            mreinfo.run_tbl = (unsigned short *)REF_TBL;
            while( err_code = mr_mrenc( &mreinfo ) ){
                switch(err_code){
                    case 1: mreinfo.enc_buf_size = BUF_SIZE; continue;
                    case 2: pix_min_err(); break;
                    case 3: pix_over_err(); break;
                }
            }

    }
    for( cnt = RTC_CNT ; cnt != 0 ; cnt-- ){
        if( mr_albit( &mreinfo )) mreinfo.enc_buf_size = BUF_SIZE;
    }
    if( mreinfo.odd_bit)  *(mreinfo.enc_buf) = mreinfo.odd_data;

}

pix_min_err(){
    exit(1);
}

pix_over_err(){
    exit(1);
}

buf_size0_err(){
    exit(1);
}
```

(4) MH方式伸長　mh_dec.c

```
#include "codec_vr.h"

#define DEC_BUF             0x84100000L
#define BUF_SIZE            0x10000
#define PRINT_BUF           0x84040000L
#define RUN_TBL             0x84020000L
#define REF_TBL             0x84020000L
#define RUN_TBL_BUF_SIZE    0x20000
#define PIXEL_NUM           1728

void main()
{
    MRDINFO mrdinfo;
    register unsigned int   c;

    mrdinfo.dec_buf = (unsigned int *)DEC_BUF;
    mrdinfo.odd_bit = 0;
    mrdinfo.pixel_data = 0;
    mrdinfo.pixel_bit = 0;
    mrdinfo.dec_buf_size = BUF_SIZE;
    mrdinfo.print_buf = (unsigned int *)PRINT_BUF;
    mrdinfo.pixel_num = PIXEL_NUM;
    mrdinfo.restart_adr = 0;

    for(;;){
        if( ( c = mr_scheol( &mrdinfo ) ) != 4 ){
            scheol_err( &mrdinfo, c );
        }
        mrdinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
        mrdinfo.run_tbl = (unsigned short *)RUN_TBL;
        if( c = mr_mhdec( &mrdinfo ) ){
            if( c == 4 ) break;
            mhdec_err( &mrdinfo, c );
        }
        mrdinfo.run_tbl = (unsigned short *)RUN_TBL;
        mr_cnvtrp( &mrdinfo );
    }

    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );

}

scheol_err( mrdinfo, c )
MRDINFO *mrdinfo;
int c;
{
    register err;
```

```
    switch( c ){
        case 1:
            do{
                mrdinfo->dec_buf_size = BUF_SIZE;
                if( mrdinfo->odd_bit != 0 ){
                    ++mrdinfo->dec_buf;
                }
            }while( ( mr_scheol( mrdinfo ) ) == 1 );
            break;
        case 5: break;
        case 12: exit(1);
    }
}

mhdec_err( mrdinfo, c )
MRDINFO *mrdinfo;
int c;
{
    register err;

    switch( c ){
        case 1:
            do{
                mrdinfo->dec_buf_size = BUF_SIZE;
                if( mrdinfo->odd_bit != 0 ){
                    ++mrdinfo->dec_buf;
                }
            }while( ( err = mr_mhdec( mrdinfo ) ) == 1 );
            break;
        case 5:
            break;
        case 6: pix_min_err(); break;
        case 7: pix_over_err(); break;
        case 8: not_encd_r(); break;
        case 9: error_code_r(); break;
    }
}

cnvtrp_err()
{
    exit(1);
}

stop_eol()
{
    exit(1);
}
stop_mrdec()
{
    exit(1);
}
stop_mhdec()
{
    exit(1);
}

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

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

not_encd_r()
{
    exit(1);
}

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

(5) MR方式伸長　mr_dec.c

```
#include "codec_vr.h"

#define DEC_BUF             0x84100000L
#define BUF_SIZE            0x10000
#define PRINT_BUF           0x84040000
#define RUN_TBL             0x84020000
#define REF_TBL             0x84020000
#define RUN_TBL_BUF_SIZE    0x10000
#define PIXEL_NUM           1728


void main()
{
    MRDINFO mrdinfo;
    register unsigned int   c;

    mrdinfo.dec_buf = (unsigned int *)DEC_BUF;
    mrdinfo.odd_bit = 0;
    mrdinfo.pixel_data = 0;
    mrdinfo.pixel_bit = 0;
    mrdinfo.run_M_len = 0;
    mrdinfo.chg_cnt = 0;
    mrdinfo.run_0_flg = 0;

    mrdinfo.dec_buf_size = BUF_SIZE;
    mrdinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
    mrdinfo.print_buf = (unsigned int *)PRINT_BUF;
    mrdinfo.ref_tbl = (unsigned short *)REF_TBL;
    mrdinfo.pixel_num = PIXEL_NUM;
    mrdinfo.restart_adr = 0;

    for(;;){
        if( ( c = mr_scheol( &mrdinfo ) ) != 4 ){
            scheol_err( &mrdinfo, c );
        }
             mrdinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
        if( !mr_getbit( &mrdinfo ) ){
            if( c = mr_mrdec( &mrdinfo ) ){
                if( c == 4) break;
                mrdec_err( &mrdinfo, c);
            }
            mrdinfo.run_tbl = mrdinfo.ref_tbl;
            mr_cnvtrp( &mrdinfo );
        }
        else
        {
            mrdinfo.run_tbl = (unsigned short *)RUN_TBL;
            mrdinfo.ref_tbl = mrdinfo.run_tbl;
            if( c = mr_mhdec( &mrdinfo ) ){
                if( c == 4) break;
                mhdec_err( &mrdinfo, c );
            }
            mrdinfo.run_tbl = mrdinfo.ref_tbl;
            mr_cnvtrp( &mrdinfo );
        }
    };
```

```
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    mr_getbit( &mrdinfo );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    mr_getbit( &mrdinfo );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    mr_getbit( &mrdinfo );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    mr_getbit( &mrdinfo );
    if( 4 != mr_scheol( &mrdinfo ))
        scheol_err( &mrdinfo, c );
    mr_getbit( &mrdinfo );

}

scheol_err( mrdinfo, c )
MRDINFO *mrdinfo;
int c;
{
    register err;

    switch( c ){
        case 1:
            do{
                mrdinfo->dec_buf_size = BUF_SIZE;
                if( mrdinfo->odd_bit != 0 ){
                    ++mrdinfo->dec_buf;
                }
            }while( ( mr_scheol( mrdinfo ) ) == 1 );
            break;
        case 5: break;
        case 12: exit(1);
    }

}

mrdec_err( mrdinfo, c )
MRDINFO *mrdinfo;
int c;
{
    register err;

    switch( c ){
        case 1:
            do{
                mrdinfo->dec_buf_size = BUF_SIZE;
                if(mrdinfo->odd_bit != 0){
                    ++mrdinfo->dec_buf;
                }
            }while( ( err = mr_mrdec( mrdinfo ) ) == 1 );
            break;
        case 5: break;
        case 6: pix_min_err(); break;
        case 7: pix_over_err(); break;
        case 8: not_encd_r(); break;
        case 9: error_code_r(); break;
        case 10: run_len_m_error(); break;
        case 11: run_len_0_error(); break;
```

```
    }
}
mhdec_err( mrdinfo, c )
MRDINFO *mrdinfo;
int c;
{
    register err;

    switch( c ){
        case 1:
            do{
                mrdinfo->dec_buf_size = BUF_SIZE;
                if( mrdinfo->odd_bit != 0 ){
                    ++mrdinfo->dec_buf;
                }
            }while( ( err = mr_mhdec( mrdinfo ) ) == 1 );
            break;
        case 5: break;
        case 6: pix_min_err(); break;
        case 7: pix_over_err(); break;
        case 8: not_encd_r(); break;
        case 9: error_code_r(); break;

    }
}

cnvtrp_err()
{
    exit(1);
}

stop_eol()
{
    exit(1);
}
stop_mrdec()
{
    exit(1);
}
stop_mhdec()
{
    exit(1);
}

pix_min_err()
{
    exit(1);
}

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

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

error_code_r()
{
    exit(1);
}

run_len_m_error()
{
    exit(1);
}

run_len_0_error()
{
    exit(1);
}
```

(6) MMR方式伸長　mmr_dec.c

```
#include "codec_vr.h"

#define DEC_BUF             0x84100000L
#define BUF_SIZE            0x10000
#define PRINT_BUF           0x84040000
#define RUN_TBL             0x84028000
#define REF_TBL             0x84020000
#define RUN_TBL_BUF_SIZE    0x10000
#define PIXEL_NUM           1728

void main()
{
    MRDINFO mrdinfo;
    register unsigned int   c;

    mrdinfo.dec_buf = (unsigned int *)DEC_BUF;
    mrdinfo.odd_bit = 0;
    mrdinfo.pixel_data = 0;
    mrdinfo.pixel_bit = 0;
    mrdinfo.run_M_len = 0;
    mrdinfo.chg_cnt = 0;
    mrdinfo.run_0_flg = 0;

    mrdinfo.dec_buf_size = BUF_SIZE;
    mrdinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
    mrdinfo.print_buf = (unsigned int *)PRINT_BUF;
    mrdinfo.ref_tbl = (unsigned short *)REF_TBL;
    mrdinfo.pixel_num = PIXEL_NUM;
    mrdinfo.restart_adr = 0;
    *(mrdinfo.ref_tbl) = PIXEL_NUM;
    *(mrdinfo.ref_tbl+1) = 0xffff;
    mrdinfo.run_tbl = (unsigned short *)RUN_TBL;

    for(;;){
        mrdinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
        if( c = mr_mrdec( &mrdinfo ) ){
            if( c == 4 ) break;
            mrdec_err( &mrdinfo, c );
        }
        mrdinfo.run_tbl = (unsigned short *)RUN_TBL;
        mr_cnvtrp( &mrdinfo );
        mrdinfo.run_tbl = (unsigned short *)REF_TBL;
        mrdinfo.ref_tbl = (unsigned short *)RUN_TBL;

        mrdinfo.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
        if( c = mr_mrdec( &mrdinfo ) ){
            if( c == 4 ) break;
            mrdec_err( &mrdinfo, c );
        }
        mrdinfo.run_tbl = (unsigned short *)REF_TBL;
        mr_cnvtrp( &mrdinfo );
        mrdinfo.run_tbl = (unsigned short *)RUN_TBL;
        mrdinfo.ref_tbl = (unsigned short *)REF_TBL;

    }
    if( 4 != mr_scheol( &mrdinfo ))
        rtc_err();
    if( 4 != mr_scheol( &mrdinfo ))
        rtc_err();
}
```

```
scheol_err( mrdinfo, c )
MRDINFO *mrdinfo;
int c;
{
    do{
        stop_eol();
    }while( ( c = mr_scheol( mrdinfo ) ) == 1 );

    if( c == 12) exit(1);
}

mrdec_err( mrdinfo, c )
MRDINFO *mrdinfo;
int c;
{
    register err;

    switch( c ){
        case 1:
            do{
                mrdinfo->dec_buf_size = BUF_SIZE;
                if( mrdinfo->odd_bit != 0 ){
                    ++mrdinfo->dec_buf;
                }
            }while( ( err = mr_mrdec( mrdinfo ) ) == 1 );
            break;
        case 5: break;
        case 6: pix_min_err(); break;
        case 7: pix_over_err(); break;
        case 8: not_encd_r(); break;
        case 9: error_code_r(); break;
    }
}

cnvtrp_err()
{
    exit(1);
}

stop_eol()
{
    exit(1);
}
stop_mrdec()
{
    exit(1);
}
stop_mhdec()
{
    exit(1);
}

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

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

not_encd_r()
{
    exit(1);
}

error_code_r(
{
    exit(1);
}

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

（7）MR方式圧縮（APIライブラリ使用）　mre_api.c

```
#include "codec_vr.h"

#define BUF_SIZE        0x10000
#define ENC_BUF         0x84100000L
#define PIXEL_BUF       0x84040000L
#define PIXEL_LINE      128
#define PIXEL_NUM       1728
#define PAGE_LINE       2376
#define RUN_TBL         0x84020000L
#define RTC_CNT         6   /* MMR:2 */
#define MODE            2   /* MH:1, MR:2, MMR:3 */
#define K_PARA          4
#define FILL_BIT        0

#define NORMAL_END      0
#define BUFF_EMPTY      1
#define PIX_MIN_ERR     2
#define PIX_OVER_ERR    3
#define PAGE_END        13


void main()
{
    MREINFO mreinfo;
    MRAPIINFO mrapiinfo;
    register unsigned int  status;


    mreinfo.enc_buf = (unsigned int *)ENC_BUF;
    mreinfo.pixel_buf = (unsigned int *)PIXEL_BUF;
    mreinfo.enc_buf_size = BUF_SIZE;
    mreinfo.pixel_num = PIXEL_NUM;
    mrapiinfo.run_len_tbl = (unsigned short *)RUN_TBL;
    mrapiinfo.page_line = PAGE_LINE;
    mrapiinfo.pixel_line = PIXEL_LINE;
    mrapiinfo.method = MODE;        /* MR mode */
    mrapiinfo.rtc_cnt = RTC_CNT;    /* MR = 6  */
    mrapiinfo.k_para = K_PARA;      /* K  = 4  */
    mrapiinfo.fill_bit = FILL_BIT;
    mrapiinfo.reset = 1;            /* reset   */

    /******************/
    /* pixel data set */
    /******************/

    while( (status = mr_encode(&mreinfo, &mrapiinfo)) != PAGE_END ){
        switch( status ) {
            case NORMAL_END :
                        /******************/
                        /* pixel data set */
                        /******************/
```

```
                        mrapiinfo.pixel_line = PIXEL_LINE;
                        break;

            case BUFF_EMPTY :mreinfo.enc_buf_size = BUF_SIZE;
                        break;

            case PIX_MIN_ERR:pix_min_err(); break;
            case PIX_OVER_ERR   :pix_over_err(); break;

        }
    }

    /* PAGE END */
    if( mreinfo.odd_bit)  *(mreinfo.enc_buf) = mreinfo.odd_data;

}


pix_min_err(){
    exit(1);
}

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

（8）MR方式伸長（APIライブラリ使用）　mrd_api.c

```
#include "codec_vr.h"

#define BUF_SIZE            0x10000
#define DEC_BUF             0x84100000L
#define PRINT_BUF           0x84040000L
#define PIXEL_LINE          128
#define PIXEL_NUM           1728
#define RUN_TBL             0x84020000L
#define MODE                2       /* MH:1, MR:2, MMR:3 */

#define NORMAL_END          0
#define BUFF_EMPTY          1
#define PIX_MIN_ERR         2
#define PIX_OVER_ERR        3
#define UNDER_ERR           6
#define OVER_ERR            7
#define NOT_COMP_ERR        8
#define UNUSUAL_ERR         9
#define RUN0_MINUS_ERR      10
#define RUN0_ERR            11
#define ERR_EOL_STATUS      12
#define PAGE_END            13


void main()
{
    MRDINFO mrdinfo;
    MRAPIINFO mrapiinfo;
    register unsigned int  status;


    mrdinfo.dec_buf = (unsigned int *)DEC_BUF;
    mrdinfo.print_buf = (unsigned int *)PRINT_BUF;
    mrdinfo.dec_buf_size = BUF_SIZE;
    mrdinfo.pixel_num = PIXEL_NUM;
    mrapiinfo.run_len_tbl = (unsigned short *)RUN_TBL;
    mrapiinfo.pixel_line = PIXEL_LINE;
    mrapiinfo.method = MODE;        /* MR mode */
    mrapiinfo.reset = 1;            /* reset   */

    while( (status = mr_decode(&mrdinfo, &mrapiinfo)) != PAGE_END ){
        switch( status ) {
            case NORMAL_END :
                        /**********************/
                        /* pixel data forward */
                        /**********************/
                        mrapiinfo.pixel_line = PIXEL_LINE;
                        break;

            case BUFF_EMPTY : mrdinfo.dec_buf_size = BUF_SIZE;
                        break;
```

```
            case UNDER_ERR      : pix_min_err(); break;
            case OVER_ERR       : pix_over_err(); break;
            case NOT_COMP_ERR   : not_encd_r(); break;
            case UNUSUAL_ERR    : error_code_r(); break;
            case RUNO_MINUS_ERR : run_len_m_error(); break;
            case RUNO_ERR       : run_len_0_error(); break;
            case ERR_EOL_STATUS : exit(1);


        }
    }
    /* PAGE END */
}


pix_min_err()
{
    exit(1);
}

pix_over_err()
{
    exit(1);
}

error_code_r()
{
    exit(1);
}

not_encd_r()
{
    exit(1);
}

run_len_m_error()
{
    exit(1);
}

run_len_0_error()
{
    exit(1);
}
```

付録B　MH/MR/MMRサンプル・ソース・リスト（AP703000-B01, AP705100-B01, AP70732-B01）

（1）MH方式圧縮　mh_enc.c

```
#include "codec810.h"

#define BUF_SIZE            0x10000
#define ENC_BUF             0x100000L
#define PIXCEL_BUF          0x40000L
#define PIXCEL_NUM          1728
#define RUN_TBL             0x20000L
#define RUN_TBL_BUF_SIZE    0x20000L
#define LINE_CNT            2376
#define RTC_CNT             6

main()
{
        struct ENCODER codec ;
        register unsigned int cnt ;
        unsigned int err_code=0;

        codec.odd_data = 0;
        codec.odd_bit = 0;
        codec.enc_buf = (unsigned int *)ENC_BUF;
        codec.pixcel_buf = (unsigned int *)PIXCEL_BUF;
        codec.enc_buf_size = BUF_SIZE;
        codec.pixcel_num = PIXCEL_NUM;
        codec.restart_adr = 0;
        codec.send_data=0x800;
        codec.send_bit=12;

        for( cnt= (LINE_CNT/4) ; cnt != 0 ; cnt-- ){
                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        codec.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
                        if( mr_coltrp( &codec ))buf_size0_err();
                        if( mr_coltrp( &codec ))buf_size0_err();
                        if( mr_coltrp( &codec ))buf_size0_err();
                        if( mr_coltrp( &codec ))buf_size0_err();
                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        codec.restart_adr = 0;
                        if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                        while( err_code = mr_mhenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                        if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
```

```
                        while( err_code = mr_mhenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                        if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                        while( err_code = mr_mhenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                        if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                        while( err_code = mr_mhenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
        }
        for( cnt=RTC_CNT ; cnt != 0 ; cnt-- ){
                if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
        }
        if( codec.odd_bit)  *(codec.enc_buf) = codec.odd_data;
}

pix_min_err(){
        exit(1);
}

pix_over_err(){
        exit(1);
}

buf_size0_err(){
        exit(1);
}
```

（2）MR方式圧縮　mr_enc.c

```
#include "codec810.h"

#define BUF_SIZE            0x10000
#define ENC_BUF             0x100000L
#define PIXCEL_BUF          0x40000L
#define PIXCEL_NUM          1728
#define RUN_TBL             0x20000L
#define RUN_TBL_BUF_SIZE    0x20000L
#define LINE_CNT            2376
#define RTC_CNT             6

main()
{
        struct ENCODER codec ;
        register unsigned int cnt ;
        unsigned int err_code = 0 ;

        codec.odd_data = 0;
        codec.odd_bit = 0;
        codec.enc_buf = (unsigned int *)ENC_BUF;
        codec.pixcel_buf = (unsigned int *)PIXCEL_BUF;
        codec.enc_buf_size = BUF_SIZE;
        codec.pixcel_num = PIXCEL_NUM;
        codec.restart_adr = 0;
        codec.send_bit=13;

        for( cnt= (LINE_CNT/4) ; cnt != 0 ; cnt-- ){
                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        codec.run_tbl_buf_size = RUN_TBL_BUF_SIZE;
                        if( mr_coltrp( &codec ))buf_size0_err();
                        if( mr_coltrp( &codec ))buf_size0_err();
                        if( mr_coltrp( &codec ))buf_size0_err();
                        if( mr_coltrp( &codec ))buf_size0_err();
                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        codec.ref_tbl = (unsigned short *)RUN_TBL;
                        codec.send_data=0x1800;
                        if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                        codec.restart_adr = 0;
                        while( err_code = mr_mhenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                        codec.send_data=0x800;
                        if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                        while( err_code = mr_mrenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
```

```
                                case 2:pix_min_err();break;
                                case 3:pix_over_err();break;
                        }
                }
                if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                while( err_code = mr_mrenc( &codec )){
                        switch(err_code){
                                case 1:codec.enc_buf_size = BUF_SIZE;
                                       continue;
                                case 2:pix_min_err();break;
                                case 3:pix_over_err();break;
                        }
                }
                if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                while( err_code = mr_mrenc( &codec )){
                        switch(err_code){
                                case 1:codec.enc_buf_size = BUF_SIZE;
                                       continue;
                                case 2:pix_min_err();break;
                                case 3:pix_over_err();break;
                        }
                }
        }
        codec.send_data=0x1800;
        for( cnt=RTC_CNT ; cnt != 0 ; cnt-- ){
                if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
        }
        if( codec.odd_bit)  *(codec.enc_buf) = codec.odd_data;
}

pix_min_err(){
        exit(1);
}

pix_over_err(){
        exit(1);
}

buf_size0_err(){
        exit(1);
}
```

（3）MMR方式圧縮　mmr_enc.c

```
#include "codec810.h"

#define BUF_SIZE            0x10000
#define ENC_BUF             0x100000L
#define PIXCEL_BUF          0x40000L
#define PIXCEL_NUM          1728
#define RUN_TBL             0x20000L
#define RUN_TBL_BUF_SIZE    0x20000L
#define REF_TBL             0x28000L
#define LINE_CNT            2376
#define RTC_CNT             2

main()
{
        struct ENCODER codec ;
        register unsigned int cnt ;
        unsigned int err_code = 0 ;

        codec.odd_data = 0;
        codec.odd_bit = 0;
        codec.enc_buf = (unsigned int *)ENC_BUF;
        codec.pixcel_buf = (unsigned int *)PIXCEL_BUF;
        codec.enc_buf_size = BUF_SIZE;
        codec.pixcel_num = PIXCEL_NUM;
        codec.restart_adr = 0;
        codec.send_data=0x800;
        codec.send_bit=12;
        codec.run_tbl = (unsigned short *)RUN_TBL;
        codec.ref_tbl = (unsigned short *)REF_TBL;
        *(codec.ref_tbl) = PIXCEL_NUM;
        *(codec.ref_tbl + 1) = 0xffff;
        codec.restart_adr = 0;

        for( cnt=(LINE_CNT/4) ; cnt != 0 ; cnt-- ){
                        codec.run_tbl_buf_size = RUN_TBL_BUF_SIZE;

                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        codec.ref_tbl = (unsigned short *)REF_TBL;
                        if( mr_coltrp( &codec ))buf_size0_err();
                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        while( err_code = mr_mrenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                        codec.run_tbl = (unsigned short *)REF_TBL;
                        codec.ref_tbl = (unsigned short *)RUN_TBL;

                        if( mr_coltrp( &codec ))buf_size0_err();
```

```
                        codec.run_tbl = (unsigned short *)REF_TBL;
                        while( err_code = mr_mrenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        codec.ref_tbl = (unsigned short *)REF_TBL;

                        if( mr_coltrp( &codec ))buf_size0_err();
                        codec.run_tbl = (unsigned short *)RUN_TBL;
                        while( err_code = mr_mrenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                        codec.run_tbl = (unsigned short *)REF_TBL;
                        codec.ref_tbl = (unsigned short *)RUN_TBL;

                        if( mr_coltrp( &codec ))buf_size0_err();
                        codec.run_tbl = (unsigned short *)REF_TBL;
                        while( err_code = mr_mrenc( &codec )){
                                switch(err_code){
                                        case 1:codec.enc_buf_size = BUF_SIZE;
                                               continue;
                                        case 2:pix_min_err();break;
                                        case 3:pix_over_err();break;
                                }
                        }
                }
                for( cnt=RTC_CNT ; cnt != 0 ; cnt-- ){
                        if( mr_albit( &codec ))codec.enc_buf_size = BUF_SIZE;
                }
                if( codec.odd_bit)  *(codec.enc_buf) = codec.odd_data;
}

pix_min_err(){
        exit(1);
}

pix_over_err(){
        exit(1);
}

buf_size0_err(){
        exit(1);
}
```

（4）MH方式伸長　mh_dec.c

```
#include "codec810.h"

#define DEC_BUF         0x100000L
#define BUF_SIZE        0x10000
#define PRINT_BUF       0x40000
#define RUN_TBL         0x20000
#define REF_TBL         0x20000
#define PIXCEL_NUM      1728


main()
{
        static struct DECODER codec;
        register unsigned int   c;
        codec.dec_buf=(unsigned int *)DEC_BUF;
        codec.odd_bit=0;
        codec.pixcel_data=0;
        codec.pixcel_bit=0;
        codec.dec_buf_size= BUF_SIZE;
        codec.print_buf=(unsigned int *)PRINT_BUF;
        codec.ref_tbl=(unsigned short *)RUN_TBL;
        codec.run_tbl=(unsigned short *)REF_TBL;
        codec.pixcel_num=PIXCEL_NUM;
        codec.restart_adr=0;

        for(;;){
                if( ( c = mr_scheol( &codec ) ) != 4){
                        scheol_err( &codec, c );
                }
                codec.run_tbl=(unsigned short *)RUN_TBL;
                if( c = mr_mhdec( &codec ) ){
                        if( c == 4) break;
                        mhdec_err( &codec, c );
                }
                codec.run_tbl = codec.ref_tbl;
                mr_cnvtrp( &codec );
        };

        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
}

scheol_err( codec, c )
struct DECODER *codec;
int c;
{
        register err;
```

```
        switch( c ){
                case 1:
                        do{
                                codec->dec_buf_size = BUF_SIZE;
                                ++codec->dec_buf;
                        }while( ( mr_scheol( codec ) ) == 1 );
                        break;
                case 5: break;
                case 12:exit(1);
        }
}
mhdec_err( codec, c )
struct DECODER *codec;
int c;
{
        register err;

        switch( c ){
                case 1:
                        do{
                                codec->dec_buf_size = BUF_SIZE;
                                ++codec->dec_buf;
                        }while( ( err = mr_mhdec( codec ) ) == 1 );
                        break;
                case 6:pix_min_err();break;
                case 7:pix_over_err();break;
                case 8:not_encd_r();break;
                case 9:error_code_r();break;
        }
}

cnvtrp_err()
{
        exit(1);
}

stop_eol()
{
        exit(1);
}
stop_mrdec()
{
        exit(1);
}
stop_mhdec()
{
        exit(1);
}

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

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

not_encd_r()
{
        exit(1);
}

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

（5）MR方式伸長　mr_dec.c

```
#include "codec810.h"

#define DEC_BUF         0x100000L
#define BUF_SIZE        0x10000
#define PRINT_BUF       0x40000
#define RUN_TBL         0x20000
#define REF_TBL         0x20000
#define PIXCEL_NUM      1728


main()
{
        struct DECODER codec;
        register unsigned int   c;
        codec.dec_buf=(unsigned int *)DEC_BUF;
        codec.odd_bit=0;
        codec.pixcel_data=0;
        codec.pixcel_bit=0;
        codec.run_M_len=0;
        codec.chg_cnt=0;
        codec.run_0_flg=0;

        codec.dec_buf_size= BUF_SIZE;
        codec.print_buf=(unsigned int *)PRINT_BUF;
        codec.ref_tbl=(unsigned short *)REF_TBL;
        codec.pixcel_num=PIXCEL_NUM;
        codec.restart_adr=0;

        for(;;){
                if( ( c = mr_scheol( &codec ) ) != 4){
                        scheol_err( &codec, c );
                }
                if( !mr_getbit( &codec ) ){
                        if( c = mr_mrdec( &codec ) ){
                                if( c == 4) break;
                                mrdec_err( &codec, c);
                        }
                        codec.run_tbl = codec.ref_tbl;
                        mr_cnvtrp( &codec );
                }
                else
                {
                        codec.run_tbl=(unsigned short *)RUN_TBL;
                        codec.ref_tbl=codec.run_tbl;
                        if( c = mr_mhdec( &codec ) ){
                                if( c == 4) break;
                                mhdec_err( &codec, c);
                        }
                        codec.run_tbl = codec.ref_tbl;
                        mr_cnvtrp( &codec );
                }
        };
```

```
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        mr_getbit( &codec );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        mr_getbit( &codec );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        mr_getbit( &codec );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        mr_getbit( &codec );
        if( 4 != mr_scheol( &codec ))
                scheol_err( &codec, c );
        mr_getbit( &codec );
}

scheol_err( codec, c )
struct DECODER *codec;
int c;
{
        register err;

        switch( c ){
                case 1:
                        do{
                                codec->dec_buf_size = BUF_SIZE;
                                ++codec->dec_buf;
                        }while( ( mr_scheol( codec ) ) == 1 );
                        break;
                case 5: break;
                case 12:exit(1);
        }

}

mrdec_err( codec, c )
struct DECODER *codec;
int c;
{
        register err;

        switch( c ){
                case 1:
                        do{
                                codec->dec_buf_size = BUF_SIZE;
                                ++codec->dec_buf;
                        }while( ( err = mr_mrdec( codec ) ) == 1 );
                        break;
                case 6:pix_min_err();break;
                case 7:pix_over_err();break;
                case 8:not_encd_r();break;
                case 9:error_code_r();break;
                case 10:run_len_m_error();break;
```

```
                case 11:run_len_0_error();break;
        }
}
mhdec_err( codec, c )
struct DECODER *codec;
int c;
{
        register err;

        switch( c ){
                case 1:
                        do{
                                codec->dec_buf_size = BUF_SIZE;
                                ++codec->dec_buf;
                        }while( ( err = mr_mhdec( codec ) ) == 1 );
                        break;
                case 6:pix_min_err();break;
                case 7:pix_over_err();break;
                case 8:not_encd_r();break;
                case 9:error_code_r();break;
        }
}

cnvtrp_err()
{
        exit(1);
}

stop_eol()
{
        exit(1);
}

stop_mrdec()
{
        exit(1);
}

stop_mhdec()
{
        exit(1);
}

pix_min_err()
{
        exit(1);
}

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

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

error_code_r()
{
        exit(1);
}

run_len_m_error()
{
        exit(1);
}

run_len_0_error()
{
        exit(1);
}
```

（6）MMR方式伸長　mmr_dec.c

```
#include "codec810.h"

#define DEC_BUF         0x100000L
#define BUF_SIZE        0x10000
#define PRINT_BUF       0x40000
#define RUN_TBL         0x28000
#define REF_TBL         0x20000
#define PIXCEL_NUM      1728

main()
{
        static struct DECODER codec;
        register unsigned int   c;
        codec.dec_buf=(unsigned int *)DEC_BUF;
        codec.odd_bit=0;
        codec.pixcel_data=0;
        codec.pixcel_bit=0;
        codec.run_M_len=0;
        codec.chg_cnt=0;
        codec.run_0_flg=0;

        codec.dec_buf_size=BUF_SIZE;
        codec.print_buf=(unsigned int *)PRINT_BUF;
        codec.ref_tbl=(unsigned short *)REF_TBL;
        codec.pixcel_num=PIXCEL_NUM;
        codec.restart_adr=0;
        *(codec.ref_tbl)=PIXCEL_NUM;
        *(codec.ref_tbl+1)=0xffff;
        codec.run_tbl=(unsigned short *)RUN_TBL;

        for(;;){
                if( c = mr_mrdec( &codec ) ){
                        if( c == 4) break;
                        mrdec_err( &codec, c );
                }
                codec.run_tbl=(unsigned short *)RUN_TBL;
                mr_cnvtrp( &codec );
                codec.run_tbl=(unsigned short *)REF_TBL;
                codec.ref_tbl=(unsigned short *)RUN_TBL;

                if( c = mr_mrdec( &codec ) ){
                        if( c == 4) break;
                        mrdec_err( &codec, c );
                }
                codec.run_tbl=(unsigned short *)REF_TBL;
                mr_cnvtrp( &codec );
                codec.run_tbl=(unsigned short *)RUN_TBL;
                codec.ref_tbl=(unsigned short *)REF_TBL;

        };
        if( 4 != mr_scheol( &codec ))
                rtc_err();
        if( 4 != mr_scheol( &codec ))
                rtc_err();
}
```

```
scheol_err( codec, c )
struct DECODER *codec;
int c;
{
        do{
                stop_eol();
        }while( ( c = mr_scheol( codec ) ) == 1);

        if( c == 12) exit(1);
}
mrdec_err( codec, c )
struct DECODER *codec;
int c;
{
        register err;

        switch( c ){
                case 1:
                        do{
                                codec->dec_buf_size = BUF_SIZE;
                                ++codec->dec_buf;
                        }while( ( err = mr_mrdec( codec ) ) == 1 );
                        break;
                case 6:pix_min_err();break;
                case 7:pix_over_err();break;
                case 8:not_encd_r();break;
                case 9:error_code_r();break;
        }
}

cnvtrp_err()
{
        exit(1);
}

stop_eol()
{
        exit(1);
}

stop_mrdec()
{
        exit(1);
}

stop_mhdec()
{
        exit(1);
}

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

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

not_encd_r()
{
        exit(1);
}

error_code_r()
{
        exit(1);
}

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

保守／廃止

★

付録Ｃ　総合索引

Ｃ.１　50音で始まる語句の索引

保守／廃止

【さ行】

【た行】

【は行】

【ま行】

【ら行】

C.2　アルファベットで始まる語句の索引

保守／廃止

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン（インフォメーションセンター）

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

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部		〒108-8001	東京都港区芝５－７－１　（日本電気本社ビル）	(03)3454−1111
中部支社	半導体第一販売部 半導体第二販売部	〒460-8525	愛知県名古屋市中区錦１－17－１　（日本電気中部ビル）	(052)222−2170 (052)222−2190
関西支社	半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540-8551	大阪府大阪市中央区城見１－４－24　（日本電気関西ビル）	(06) 945−3178 (06) 945−3200 (06) 945−3208

北海道支社	札　幌	(011)231−0161	宇都宮支店	宇都宮	(028)621−2281	北陸支社	金　沢	(076)232−7303
東北支社	仙　台	(022)267−8740	小山支店	小　山	(0285)24−5011	富山支店	富　山	(0764)31−8461
岩手支店	盛　岡	(019)651−4344	甲府支店	甲　府	(0552)24−4141	福井支店	福　井	(0776)22−1866
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	静岡支社	静　岡	(054)254−4794	神戸支社	神　戸	(078)333−3854
長岡支店	長　岡	(0258)36−2155	立川支社	立　川	(042)526−5981,6167	中国支社	広　島	(082)242−5504
水戸支店	水　戸	(029)226−1717	埼玉支社	大　宮	(048)649−1415	鳥取支店	鳥　取	(0857)27−5311
土浦支店	土　浦	(0298)23−6161	千葉支社	千　葉	(043)238−8116	岡山支店	岡　山	(086)225−4455
群馬支店	高　崎	(027)326−1255	神奈川支社	横　浜	(045)682−4524	松山支店	松　山	(089)945−4149
太田支店	太　田	(0276)46−4011	三重支店	津	(059)225−7341	九州支社	福　岡	(092)261−2806

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］μSAP30100-B01, μSAP703000-B01, μSAP705100-B01, μSAP70732-B01　ユーザーズ・マニュアル
（U10338JJ4V0UM00（第４版））

［お名前など］（さしつかえのない範囲で）

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

１．ご評価（各欄に○をご記入ください）

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

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

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

４．ご意見，ご要望

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

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

NEC半導体テクニカルホットライン
FAX：（044）548-7900

キリトリ