(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2007-172818
(P2007-172818A)
(43)公開日　平成19年7月5日(2007.7.5)

(51)Int.Cl.		F I			テーマコード（参考）
G11B 20/18	(2006.01)	G11B	20/18	532E	5J065
H03M 13/15	(2006.01)	H03M	13/15		
H03M 13/29	(2006.01)	H03M	13/29		
		G11B	20/18	512C	
		G11B	20/18	572G	

審査請求 未請求　請求項の数 30　ＯＬ　（全 29 頁）　最終頁に続く

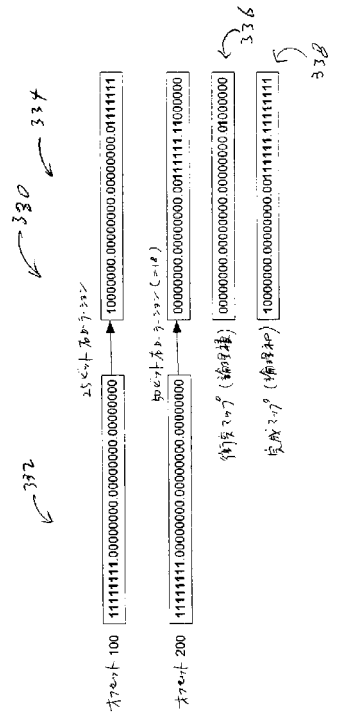
(21)出願番号　特願2006-341788 (P2006-341788)
(22)出願日　平成18年12月19日 (2006.12.19)
(31)優先権主張番号　11/313,438
(32)優先日　平成17年12月20日 (2005.12.20)
(33)優先権主張国　米国 (US)

(71)出願人　591179352
クウォンタム・コーポレイション
ＱＵＡＮＴＵＭ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国、９５１１０　カリフォルニア州、サン・ノゼ、テクノロジー・ドライブ、１６５０、スイート・８００
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄
(74)代理人　100083703
弁理士　仲村　義平
(74)代理人　100096781
弁理士　堀井　豊

最終頁に続く

(54)【発明の名称】インタリーブされたパリティチェックおよびリードソロモン符号を用いる誤り訂正アルゴリズム

(57)【要約】
【課題】データストリームの誤りを訂正するための方法および装置を提供する。
【解決手段】１つの局面では、誤り訂正プロセスは、少なくとも２つの別個の種類のデータストリームの誤りを訂正するよう作動可能であり、それぞれは異なった誤り訂正スキームを利用する。誤り訂正プロセスは、誤り訂正についてリードソロモン符号および誤り検出符号（ＥＤＣ）の組合せを利用する。このプロセスは、複数のデータブロックの誤りを特定して誤りを訂正するよう試みるために、リードソロモン符号を用いるステップを含む。するとＥＤＣマップが生成され、このＥＤＣマップは、リードソロモン符号によって訂正されない誤りを含む、対応するデータブロック内の誤りの位置を特定する。
【選択図】図３０



【特許請求の範囲】
【請求項１】
複数の誤り検出符号（ＥＤＣ）を含む、符号化されたデータのストリームを受取るように作動可能な誤り訂正装置であって、前記誤り訂正装置は、
符号化されたデータの前記ストリームに由来する二次元データアレイの第１の次元の誤りを検出するために前記複数のＥＤＣを用いるよう、かつ前記二次元データアレイの第２の次元の符号語を訂正するためにリードソロモン符号を用いるよう作動可能なデコーダを含み、前記デコーダはさらに、リードソロモン符号からＥＤＣのうちの少なくとも１つの中の対応するビットへ符号語をマッピングするよう作動可能である、誤り訂正装置。
【請求項２】
前記リードソロモン符号の記号サイズは、前記ＥＤＣのうちの少なくとも１つの記号サイズよりも小さい、請求項１に記載の誤り訂正装置。
【請求項３】
前記誤り訂正装置はコントローラの一部として構成され、前記コントローラは、データ記憶媒体に記録されたデータを検出するためのトランスデューサに結合される、請求項１に記載の誤り訂正装置。
【請求項４】
前記トランスデューサは磁気テープドライブヘッドを含み、前記データ記憶媒体は磁気テープを含む、請求項３に記載の誤り訂正装置。
【請求項５】
前記データ記憶媒体は、光ディスク、磁気ディスク、光学テープ、および磁気テープからなる群から選択される、請求項２に記載の誤り訂正装置。
【請求項６】
前記ＥＤＣのうち少なくとも１つは、ｇ（ｘ）＝x^n＋１の形を有する原始多項式によって記述され、ここでｎはＥＤＣ記号内のビット数である、請求項１に記載の誤り訂正装置。
【請求項７】
前記ＥＤＣの生成多項式はＧ（Ｘ）＝Ｘ＋α^sの形を有し、ここでαは前記原始多項式の根であり、ｓは０からｎ－１のいずれかの整数である、請求項６に記載の誤り訂正装置。
【請求項８】
前記デコーダは、前記リードソロモン符号からＥＤＣ残余へ前記符号語を関連付けるためのＥＤＣマップを生成するよう作動可能であり、前記ＥＤＣマップは、前記リードソロモン符号語内の変化する記号により影響を受け得る前記ＥＤＣ残余内の前記ビットについては２進法の１、および他の全てのビットについては２進法の０から構成される、請求項１に記載の誤り訂正装置。
【請求項９】
前記ＥＤＣマップは、巡回ビットローテーションまたは他のマッピング演算を実行することによって正規化され、そのためビットマスク内の前記ビットは前記ＥＤＣ残余内の対応するビットに直接マッピングする、請求項８に記載の誤り訂正装置。
【請求項１０】
前記デコーダはＥＤＣマップを用いるように作動可能であり、訂正不能または信頼できないように訂正されたと判断された顕著なリードソロモン符号語について誤りパターンを判断するよう構成される、請求項１に記載の誤り訂正装置。
【請求項１１】
２つ以上のＥＤＣマップが生成され、全ての前記ＥＤＣマップのビットワイズの論理和が用いられて、全ての前記顕著なリードソロモン符号語によって影響を受ける前記ＥＤＣ残余内の前記ビットを決定する、請求項９に記載の誤り訂正装置。
【請求項１２】
２つ以上のＥＤＣマップが生成され、前記ＥＤＣマップ内でビットワイズの論理積およ

び論理和演算が用いられて、２つ以上のリードソロモン符号語が前記ＥＤＣ残余内で同じビットに影響を及ぼすことができるか否かを判断する、請求項９に記載の誤り訂正装置。
【請求項１３】
前記複数のＥＤＣからの第２のＥＤＣは誤り検出のためにのみ用いられ、前記複数のＥＤＣからの第１のＥＤＣからの訂正が信頼できるか否かを判断することができる、請求項１に記載の誤り訂正装置。
【請求項１４】
前記第２のＥＤＣは巡回冗長検査（ＣＲＣ）を含む、請求項１３に記載の誤り訂正装置。
【請求項１５】
前記第２のＥＤＣはリードソロモン符号を含む、請求項１３に記載の誤り訂正装置。
【請求項１６】
データ誤り訂正の方法であって、
複数のデータブロックにおける誤りを特定して前記誤りを訂正するよう試みるためにリードソロモン符号を用いるステップと、
前記データブロックのいずれかに誤りがあるか否かを判断するために巡回冗長検査を用いるステップと、
前記リードソロモン符号によって訂正されない前記データブロックの誤りを訂正するために各データブロックの誤り検出符号（ＥＤＣ）を用いるステップとを含む、方法。
【請求項１７】
誤りを訂正するために各データブロックのＥＤＣを用いる前記ステップは、ＥＤＣマップを生成するステップをさらに含み、前記ＥＤＣマップは、前記リードソロモン符号によって訂正されない誤りを含む対応するデータブロック内の前記誤りの前記位置を特定する、請求項１６に記載の方法。
【請求項１８】
誤りを訂正するために各データブロックのＥＤＣを用いる前記ステップは、前記リードソロモン符号によって訂正されない前記データブロックの少なくとも１つの誤りを訂正するために、前記ＥＤＣから生成されるマップを用いるステップをさらに含む、請求項１６に記載の方法。
【請求項１９】
ＥＤＣマップを生成する前記ステップは、
各顕著なリードソロモン符号語について１組のＥＤＣマップを生成するステップを含み、前記ＥＤＣマップは、前記リードソロモン符号語内の変化する記号により影響を受け得る前記ＥＤＣ残余内の前記ビットについては２進法の１、および、マッピング演算を実行することによって前記ＥＤＣマップを正規化する他の全てのビットについては２進法の０から構成され、そのため、前記ビットマスク内の前記ビットは前記ＥＤＣ残余内の対応するビットに直接マッピングする、請求項１７に記載の方法。
【請求項２０】
前記マッピング演算は、巡回ビットローテーションを含む、請求項１９に記載の方法。
【請求項２１】
前記ＥＤＣの少なくとも１つは、$g(x) = x^n + 1$の形を有する原始多項式によって記述され、ここでｎはＥＤＣ記号内のビット数である、請求項１８に記載の方法。
【請求項２２】
前記ＥＤＣの生成多項式は$G(X) = X + \alpha^s$の形を有し、ここでαは前記原始多項式の根であり、ｓは０からｎ－１のいずれかの整数である、請求項２１に記載の方法。
【請求項２３】
請求項１８に記載の前記方法を実行するよう作動可能な、テープドライブ。
【請求項２４】
テープドライブであって、
磁気テープ上のデータを読取るよう作動可能な磁気テープドライブヘッドを含み、前記

データは複数のデータブロックを含み、
前記磁気テープドライブヘッドに電気的に接続されるコントローラを含み、前記コントローラは、前記データの誤りを訂正するために各データブロックの誤り訂正符号（ＥＣＣ）フィールドを用いるよう作動可能であり、前記コントローラはさらに、前記各データブロックの誤りについて検査する際に前記各データブロックの巡回冗長検査（ＣＲＣ）フィールドを用いるよう作動可能である、テープドライブ。
【請求項２５】
前記データブロックの各々は、ＥＣＣフィールドまたはＣＲＣフィールドのいずれかを有し、前記ＥＣＣフィールドおよび前記ＣＲＣフィールドは前記データブロックにおいて同じ位置を占め、前記コントローラはさらに前記データブロックの誤りを訂正するためにＥＤＣフィールドを用いるよう作動可能である、請求項２４に記載のテープドライブ。
【請求項２６】
前記コントローラはさらに、各データブロックの前記ＥＤＣフィールドに基づいてマップを生成するよう作動可能であり、前記マップは、前記対応するデータブロックにおいて誤りの前記位置を特定するために前記コントローラによって用いられるよう構成される、請求項２５に記載のテープドライブ。
【請求項２７】
前記コントローラは、前記複数のデータブロックにおいて誤りを訂正するためにリードソロモン符号を用いるよう作動可能であり、前記コントローラはさらに、前記リードソロモン符号からＥＤＣ残余へ符号語を関連付けるためのＥＤＣマップを生成するよう作動可能である、請求項２５に記載のテープドライブ。
【請求項２８】
前記ＥＤＣマップは、前記リードソロモン符号語内の変化する記号により影響を受け得る前記ＥＤＣ残余内の前記ビットについては２進法の１、および他の全てのビットについては２進法の０から構成される、請求項２７に記載のテープドライブ。
【請求項２９】
磁気テープドライブであって、
複数のデータブロックの誤りを特定して前記誤りを訂正するよう試みるためにリードソロモン符号を用いるよう、前記データブロックのいずれかに誤りがあるか否かを判断するために巡回冗長検査を用いるよう、かつ、前記リードソロモン符号によって訂正されない前記データブロックの誤りを訂正するために各データブロックのＥＤＣを用いるよう作動可能なコントローラを含み、前記コントローラはさらにＥＤＣマップを生成するよう作動可能であり、前記ＥＤＣマップは、前記リードソロモン符号によって訂正されない前記対応するデータブロック内の前記誤りの前記位置を特定するために前記コントローラによって用いられる、磁気テープドライブ。
【請求項３０】
前記コントローラはさらに、前記リードソロモン符号によって訂正されない前記データブロックの少なくとも１つの誤りを訂正するために、前記ＥＤＣから生成されるマップを用いるよう作動可能である、請求項２９に記載の磁気テープドライブ。
【発明の詳細な説明】
【技術分野】
【０００１】
技術分野
本発明は、一般に誤り訂正符号の分野に関する。本発明の一局面において、本願明細書で開示される機器および方法は、磁気テープドライブにおいて実現することができる。
【背景技術】
【０００２】
背景
データ記憶媒体に記憶されるデータの密度が増加するにつれて、媒体に記憶されたデータの完全性を確認し、書込みおよび／または読取りプロセス中に起こり得る誤りを訂正す

るために、より高度な誤り訂正方法が必要である。加えて、データ記憶装置がより小さくなり、より速い速度で作動するにつれて、データ記憶媒体からデータを読取る際の誤り率も増加する。

【０００３】

例えば、より小さい読取り装置を有する改良されたテープドライブが古い世代の磁気テープを読取るために利用されると、読取り装置の寸法がより小さく、および／または作動速度がより速いために、誤り率が著しく高まり得る。加えて、古い世代のテープは、より新しい世代のテープよりもさらに原始的な誤り訂正符号を利用していることがある。その結果、新世代のテープドライブは、古い世代のテープと後方互換性がないことがある。

【発明の開示】

【発明が解決しようとする課題】

【０００４】

したがって、新世代の誤り訂正符号を実現すると同時に、古いフォーマットで保存されたデータが新世代のシステムによって検索されて利用されるよう、古い世代の誤り訂正符号を活用することができる、誤り訂正プロセスが必要とされる。

【課題を解決するための手段】

【０００５】

発明の概要

データストリームの誤りを訂正するための方法および機器が、本願明細書において記載される。一つの局面では、誤り訂正装置は、少なくとも２つの別個の種類のデータストリームにおける誤りを訂正するように作動可能であり、各々は異なった誤り訂正スキームを用いる。一例において、誤り訂正装置は、複数の誤り検出符号（ＥＤＣ）を含む、符号化されたデータのストリームを受取るように作動可能であり、この誤り訂正装置は、符号化されたデータのストリームから構成される２つの二次元データアレイの、第１の次元の誤りを検出するために複数のＥＤＣを用いるように、かつ、この２つの二次元データアレイの第２の次元の符号語を訂正するためにリードソロモン符号を用いるように作動可能なデコーダを含む。コントローラはさらに、リードソロモン符号から１つ以上のＥＤＣ内の対応するビットへ符号語をマッピングするよう作動可能である。１つの変形例において、リードソロモン符号の記号サイズは、ＥＤＣの少なくとも１つの記号サイズよりも小さい。別の変形例において、誤り訂正装置はコントローラの一部として構成される。コントローラは、データ記憶媒体に記録されたデータを検出するためのトランスデューサに結合される。１つの特定の適用例において、トランスデューサは磁気テープドライブヘッド（例えば磁気抵抗型テープヘッド）を含み、データ記憶媒体は磁気テープを含む。他の適用例において、誤り訂正装置は、光ディスク、磁気ディスクまたは光学テープから読取られたデータを処理するよう構成され得る。

【０００６】

一実現例では、デコーダは、符号語をリードソロモン符号からＥＤＣ残余に関連付けるためのＥＤＣマップを生成するように作動可能であり、ＥＤＣマップは、リードソロモン符号語内の変化する記号により影響を受け得るＥＤＣ残余内のビットについては２進法の１、および他の全てのビットについては２進法の０から構成される。ＥＤＣマップは、巡回ビットローテーションまたは他のマッピング演算を実行することにより正規化され、その結果、ビットマスク内のビットがＥＤＣ残余内の対応するビットに直接マッピングする。１つの変形例において、２つ以上のＥＤＣマップが生成され、全てのＥＤＣマップのビットワイズ論理和が用いられて、全ての顕著なリードソロモン符号語によって影響を受けるＥＤＣ残余のビットを決定する。別の変形例において、２つ以上のＥＤＣマップが生成され、ＥＤＣマップ内でビットワイズ論理積演算および論理和演算が用いられて、２つ以上のリードソロモン符号語がＥＤＣ残余内の同じビットに影響を及ぼすことができるか否かを決定する。

【０００７】

いくつかの変形例においては、ＥＤＣのうち少なくとも１つは、$g(x)=x^n+1$の形

を有する原始多項式によって記述され、ここでｎはＥＤＣ記号内のビット数である。ＥＤＣの生成多項式はＧ(Ｘ)＝Ｘ＋α^{s}の形をとることができ、ここでαは原始多項式の根であって、ｓは０からｎ－１のいずれかの整数である。

【０００８】

別の局面においては、リードソロモン符号およびＥＤＣを用いた誤り訂正のための方法が記載される。一例において、その方法は、複数のデータブロックにおける誤りを特定して誤りを訂正するよう試みるためにリードソロモン符号を用いるステップと、データブロックのいずれかに誤りがあるか否かを判断するために巡回冗長検査を用いるステップと、リードソロモン符号によって訂正されないデータブロックの誤りを訂正するために各データブロックの誤り検出符号（ＥＤＣ）を用いるステップとを含む。誤りを訂正するため各データブロックのＥＤＣを用いるためには、ＥＤＣマップが生成され、ＥＤＣマップは、リードソロモン符号によって訂正されない誤りを含む、対応するデータブロック内の誤りの位置を特定する。

【０００９】

１つの変形例において、ＥＤＣマップは、各顕著なリードソロモン符号語について一組のＥＤＣマップを作成することによって生成され、ＥＤＣマップは、リードソロモン符号語内の変化する記号により影響を受け得るＥＤＣ残余内のビットについては２進法の１、および他の全てのビットについては２進法の０から構成される。ＥＤＣマップは次にマッピング演算を実行することによって正規化され、その結果、ビットマスク内のビットはＥＤＣ残余内の対応するビットに直接マッピングする。一例において、マッピング演算は巡回ビットローテーションを含む。

【００１０】

本発明のこれらの、および他の実施例、特徴および利点は、添付の簡単に説明される図面と関連して下記の本発明のより詳細な説明を考慮すると、当業者にはより明らかになるであろう。

【発明を実施するための最良の形態】

【００１１】

発明の詳細な説明

下記の詳細な説明は図面を参照して読まれるべきであり、同一の参照番号は異なる図面の全体にわたって同様の要素を指す。図面は必ずしも縮尺通りではないが、選択的な実施例を表し、本発明の範囲を限定することを意図しない。詳細な説明は、例として、限定するためではなく、本発明の原理を示す。この説明は明らかに、当業者が本発明を行い、使用することを可能にし、本発明を実施する最良の形態であると現在考えられるものを含む、本発明のいくつかの実施例、適合、変形例、代替および使用例を記載する。

【００１２】

本願明細書においては、本発明のさまざまな局面を示すため、本願明細書において開示される誤り訂正方法および装置を用いるための適用例として磁気テープドライブが用いられる。当業者は、本願明細書における開示に照らし、本願明細書に開示される方法および機器が、データ記憶媒体から検索されるデータの完全性を確実にするために誤り訂正符号（ＥＣＣ）を用いる、さまざまなメモリ記憶コンポーネントまたはデータ転送装置において実現され得ることを理解する。例えば、本願明細書に開示される方法および機器は、データ確認および保護を与えるために、さまざまな他のデータ記憶システム（例えばハードディスク、光学ドライブなど）で実現されることもできる。別の例では、データバッファリングプロセスにおいてＥＣＣを実行するネットワークルータが、効率的なＥＣＣ利用を達成するために、本願明細書に開示される方法および機器を活用してもよい。さらに別の例において、本願明細書に開示される方法および機器は、記憶媒体から検索されるデータを検査して訂正するために、メモリコントローラにおいて用いられ得る。この方法および機器はまた、２つ以上のデータ符号化フォーマットがデータ記憶システムによってサポートされるよう、後方互換性を与えるように構成されてもよい。

【００１３】

この明細書および添付の請求の範囲において用いられているように、単数形（"a", "an", "the"）は、文脈が明らかに逆を示す場合以外は、複数の指示対象をも含むことにも注意されなければならない。したがって、例えば、「トランスデューサ（a transducer）」は単一のトランスデューサまたはトランスデューサの組合せを意味するよう意図され、「電気信号（an electrical signal）」は１つ以上の電気信号またはその変調を意味するよう意図される。加えて、本願明細書において用いられるＥＤＣは、ＥＤＣ残余、冗長ブロック、パリティブロック、当業者に周知の誤り検出符号の変形例、およびそれらの改良を含むが、これらに限定されない。本願明細書において用いられるリードソロモン符号は、標準リードソロモン符号、当業者に周知のリードソロモン符号の変形例、およびそれらの改良を含む。

【００１４】

図１を参照すると、１つの適用例において、誤り訂正装置１０は、磁気テープから読取られたデータストリームの誤りを訂正するための、磁気テープドライブ１４内のコントローラ１２の一部として実現される。誤り訂正装置は、誤り訂正アルゴリズムを実行するよう作動可能なデコーダを含む。磁気テープドライブヘッド１８は磁気テープ上のデータを読取るよう作動可能であり、データは複数のデータブロックを含む。コントローラは磁気テープドライブヘッド１８に電気的に接続され、コントローラは、データの誤りを訂正するために、各データブロックにおいて誤り訂正符号（ＥＣＣ）フィールドを使用するよう作動可能である。１つの構成において、コントローラは、各データブロックの誤りを検査するために、データブロックの各々において巡回冗長検査（ＣＲＣ）フィールドを用いるよう作動可能である。

【００１５】

１つの変形例において、誤り訂正装置は、２つの個別のフォーマット（例えば、一方はＥＣＣフィールドを含む符号化スキームを用いて書込まれ、他方はＣＲＣフィールドを含む符号化スキームを用いて書込まれた、２つの磁気データ記憶テープ）に保存されたデータを復号化するよう構成され、データブロックは、ＥＣＣフィールドまたはＣＲＣフィールドのいずれかを有し、ＥＣＣフィールドおよびＣＲＣフィールドはデータブロックにおいて同じ位置を占める。コントローラは、データブロックの誤りを訂正するためにＥＤＣフィールドを用いるよう、さらに作動可能である。１つの変形例において、コントローラは、各データブロックにおいてＥＤＣフィールドに基づいてマップを生成するように作動可能であり、マップは、対応するデータブロックにおいて誤りの位置を特定するためにコントローラによって用いられるよう構成される。一例において、コントローラは、複数のデータブロックにおいて誤りを訂正するためにリードソロモン符号を用いるよう、かつリードソロモン符号からＥＤＣ残余へ符号語を関連付けるためのＥＤＣマップを生成するように作動可能である。一実現例では、ＥＤＣマップは、リードソロモン符号語内の変化する記号により影響を受け得るＥＤＣ残余内のビットについては２進法の１、および他の全てのビットについては２進法の０から構成される。

【００１６】

典型的な誤り訂正プロセスが下記に示される。この例では、誤り訂正プロセスはテープドライブのコントローラで実現される。より強力な形の誤り訂正プロセス（すなわち二次元のリードソロモン符号）が用いられ、それによりテープドライブは、従来のテープドライブの多くと比較してより高性能点で作動することが可能になる一方、より高いビット誤り率を有する媒体を許容することとなる。この誤り訂正プロセスは、古い世代のテープドライブより小さな読取り装置の使用が可能になる一方で、例えばＣＲＣで符号化されたものなど以前のフォーマットで書込まれたテープの読取りをも可能にする。

【００１７】

この例で用いられるＥＣＣは二次元の符号であって、積符号と呼ばれる。第１の次元はブロックであり、誤り検出または誤り訂正については、６４ビットのＣＲＣ、または１６ビットのリードソロモン符号を用いる。第２の次元はエンティティであり、一群のデータブロックおよびＥＣＣブロックを含む。テープドライブは、データをテープに書込む際に

エンベロープと呼ばれる構成を用い、その目的は全てのチャンネルにわたってエンティティをインタリーブし、そのため境界のチャンネルへの露出を最小化することである。テープドライブがホストからデータのストリームを受取るときに、テープドライブのメモリコントローラは、このデータをデータブロックと呼ばれる基本単位に入れる。デコーダのＥＣＣエンジンは次に、破損した、または失われたあらゆるデータブロックの再建を助けるＥＣＣブロックを生成する。

【００１８】

データブロックは、さまざまな制御フィールドにユーザデータを含む。この例示的なテープドライブのブロックのレイアウト２０の概要が図２に示される。テープドライブは、ホストから受取る情報を用いて各データブロックのユーザデータ２２セクションを生成する。ユーザデータセクションに続くのは、ＥＤＣ（誤り検出符号）フィールド２４である。これは、前のユーザデータ領域のチェックサムを含む、３２ビットのフィールドである。訂正アルゴリズムは訂正確認のためにＥＤＣを用い、またブロックデータ自体を訂正するために用いられてもよい。ＥＤＣを計算するためにドライブによって用いられるアルゴリズム３０が図３に示される。ＥＤＣを確認することは、同じアルゴリズムを実行することであるが、ＥＤＣフィールドを通して続けることである。これは残余を生成する。残余がゼロである場合、ＥＤＣ検査は成功したことになる。残余が非ゼロである場合、ブロックのいずれかの部分が破損を含む。

【００１９】

ＣＦ１（制御フィールド１）２６は１６バイト長であって、このブロックを論理的に特定するのに用いられ得る情報を含む。ＣＦ２（制御フィールド２）２８は、テープ上および誤り訂正構成内での物理的な位置を特定するのに用いられ得る情報を含む。これは、このブロック内の欠陥を訂正するために用いられる冗長情報を含む、６４バイトのフィールドである。ハードウェアはこのデータがテープへ出ていくに従って、これをブロックに追加する。訂正ハードウェアは、ブロックを訂正するためにこれらのバイトを使用した後、読取り動作中にバイトを取除く。ＥＣＣ１ハードウェアは、ブロック内で最大１６の破損記号を訂正することができる。この場合、各記号は１６ビット幅である。ＥＣＣ２ハードウェア（ＲＥＣと呼ばれる）は、一組のデータブロックを受けて冗長情報を生成する。ＲＥＣハードウェアは、「部分シンドローム生成を伴うセルを用いた誤り訂正デコーダ（Error Correction Decoder Using Cells with Partial Syndrome Generation）」と題されて２００５年８月２５日に公開された、米国特許出願公開番号第２００５／０１８８２９３Ａ１号において詳細に記述され、本願明細書においてその全体が引用にて援用される。この新しく生成されたブロックはＥＣＣブロック４０であり、その唯一の目的はあらゆる破損したデータを訂正するのを助けることである。そのフォーマットは図４に示される。ＥＣＣ２ブロックのためのＣＦ２　４２およびＥＣＣ１　４４のフィールドは、データブロックのためのそれらと同じである。

【００２０】

この例において、全ての逆読みフォーマットについてのブロックレイアウトは、６４ビットのＣＲＣがＥＣＣ１フィールド２９を置換することを除いては、クウォンタム（Quantum）ＤＬＴ－Ｓ４テープドライブのネイティブフォーマットのそれと実質的に同じである。ＣＲＣは誤り検出を実行できるだけであって訂正能力はない。このテープドライブの例示的な誤り訂正プロセスにもサポートされる、古いフォーマットのデータ５０およびＥＣＣ５２ブロックの両方のレイアウトが図５に示される。

【００２１】

クウォンタムＤＬＴ－Ｓ４のＥＣＣ２アルゴリズムは、エンティティと呼ばれている構成を用いることにより、ブロック全体の誤りから保護する。エンティティは、破損したデータブロックを再建するために用いられる１６個のＥＣＣブロックを伴った、１個から１２個のデータブロックの集合である。エンティティ６０のレイアウトは図６に示される。ラベル「符号語」を有する縦の長方形に注意されたい。各エンティティはこれらの符号語を１２，２８８個含み、それらは最も左側からＣＦ１の端部まで延々と及ぶ。ＣＦ２フ

ィールドおよびＣＲＣ（またはＥＣＣ１）フィールドは含まれない。各符号語は、エンティティ内の各ブロックから１つの記号（この文脈ではバイト）を含む。この例では、符号語は全てのブロック内で同じオフセットにある記号の集合である。ＥＣＣ２アルゴリズムは、訂正動作を実行する際、符号語を独立に処理する。

【００２２】

市販のテープドライブにおいて実現される、以前のフォーマット（例えばクウォンタムＳＤＬＴ３２０、クウォンタムＳＤＬＴ６００）は、ブロック数が少ないことを除いては、同じ基本エンティティレイアウトを用いる。特に、１個から１６個のデータブロックおよび４個のＥＣＣブロックを含む、逆読み互換性（ＢＲＣ）フォーマットである。テープ上のこの相対的なブロックのレイアウトについては後述する。データを書込む際、このテープドライブはエンベロープと呼ばれる基本単位を用いる。エンベロープは、散乱パターンの全てのチャンネルにわたってインタリーブされた、１個から８個のエンティティを含む。テープドライブは、ストライプごとにエンベロープを書込む。ストライプは、全てのチャンネルによって同時に書かれた、ブロックの縦のグループである。この例示的なドライブを用いて磁気テープ上に書込まれたエンベロープ７２の例示的レイアウトが図７に示される。各長方形はテープ上のブロックを表す。陰影をつけられた長方形は、第１のエンティティのための散乱パターンを示す。他の７個のエンティティの各々は同じパターンを有するが、開始位置は異なる。名目上のエンベロープは８個のエンティティを保持し、６４個のストライプに及ぶ。

【００２３】

古いフォーマットのエンベロープのレイアウト８２の例が図８に示される。この例は、クウォンタムＳＤＬＴ６００テープドライブにおいて利用されるフォーマットを示す。この特定のフォーマットは、結合されたチャネル（coupled channel）の概念を用い、これは、以前の製品と同じ数の論理チャネルを保つ一方でテープ上の物理チャネル数を倍にする、簡単なやり方である。各エンベロープは８個のエンティティを含み、最大サイズのエンティティは２０個（１６個のデータおよび４個のＥＣＣ）のブロックを含む。この例示的なテープドライブにサポートされる別のフォーマット（すなわちクウォンタムＳＤＬＴ２２０／３２０）が図９に示される。ＳＤＬＴ２２０／３２０フォーマットはＥＣＭＡ仕様（ＥＣＭＡ－３２０）として公開される。図９において、エンベロープ９２の各々は、名目上、２０個のストライプおよび８個のエンティティを含む。

【００２４】

誤り訂正プロセスは、読取りプロセスのサブセットである。読取りプロセス１００を示すフローチャートが図１０に示される。ドライブはいくつかのデータを読取る。データが不良な場合、ドライブはその訂正を試みる。訂正が失敗した場合、ドライブはヘッドを再位置決めし、データを再度読取ることを試みる。これがあまりに何度も起ると、ドライブは断念し、読取困難な誤り（Ｈａｒｄ　Ｒｅａｄ　Ｅｒｒｏｒ）（ＨＲＥ）信号を出す。訂正プロセス１１０の概要を示すダイヤグラムが図１１に示される。

【００２５】

訂正プロセスは、３つの基本アルゴリズムから選択する。消失訂正（Ｅｒａｓｕｒｅ　Ｃｏｒｒｅｃｔｉｏｎ）、拡大訂正（Ｅｘｔｅｎｄｅｄ　Ｃｏｒｒｅｃｔｉｏｎ）および全数訂正（Ｅｘｈａｕｓｔｉｖｅ　Ｃｏｒｒｅｃｔｉｏｎ）である。拡大訂正および全数訂正は、別の「反復」を実行する能力を有する。これは、訂正アルゴリズムがいくつかのデータを回復したが、残るデータを得るために別の訂正を試みる必要がある場合に該当する。

【００２６】

訂正プロセスの第１のステップは、「訂正タイプの決定」１２０である。図１２はこのプロセスを概説する。このアルゴリズムは、問題のエンティティを調べることによって開始し、失われたブロックの個数（消失数）、部分的に回復されたブロックの個数（ＥＣＣ１またはＣＲＣが不良だがそれ以外は良い）、および良いブロックの個数を決定する。数量ＴｏｔａｌＢａｄは、消失したブロックおよび部分的に回復されたブロック（すなわち

良くないものすべて）の和である。ＴｏｔａｌＢａｄがこのフォーマットからのＥＣＣブロックの最大個数より少ないかまたはこれと等しい場合、次に装置は、最適化されたエンティティ消失訂正（Ｅｎｔｉｔｙ　Ｅｒａｓｕｒｅ　Ｃｏｒｒｅｃｔｉｏｎ）を用いることができる。特例として、装置は少なくとも１つの不良なデータブロックがあることを再確認する。そうでない場合には、訂正を実行する必要はない。ＴｏｔａｌＢａｄがＥＣＣブロックの最大個数より多い場合は、装置は拡大訂正アルゴリズムを用いる必要がある。これは、消失した（すなわち完全に失われた）ブロックの個数（ＥｒａｓｕｒｅＣｎｔ）がＥＣＣブロックの最大個数より少ない場合にのみ可能である。一実現例では、あまりに多くの消失がある場合、訂正がうまく実行され得ないために、装置は訂正プロセスを中止する。

【００２７】

全ての訂正アルゴリズムにおいて最も基本となるのは、エンティティ消失訂正アルゴリズムである。このアルゴリズムは、最大でも１６個の破損ブロック（またはいくつかの市販のテープドライブにおいて実現されたいくつかの以前のフォーマットであれば最大でも４個）がある場合に有効であり得る。この例において、装置は既知の値と同程度に未知の値を有する、１組の単純な線形方程式を有する。ＥＣＣ２ハードウェア補助（ＲＥＣ）は、これらの線形方程式を解くためにマトリクス乗算演算を高速で実行することができる。訂正プロセスは、訂正マトリクスを生成し、失われたデータブロックを訂正するのにハードウェアがこのマトリクスを用いるようにプログラムすることを含む。

【００２８】

消失訂正を実行する一方で、ハードウェアは、新しく生成される全てのブロックにＥＤＣ検査を実行する。訂正が終わると、ファームウェアが、新しく生成された各データブロックのＥＤＣ残余を検査する。ＥＤＣ検査により、いずれかのデータブロックが誤りを有することが示された場合、消失訂正アルゴリズムは失敗し、誤りはエンティティが訂正可能でないことを示す。図１３は、消失訂正１３０の全プロセスを示す図である。

【００２９】

拡大訂正アルゴリズム（部分的訂正としても知られる）は、比較的多数の部分的に回復されたブロック（不良なＥＣＣ１またはＣＲＣ６４を有するもの）を含むエンティティを訂正することができる。各符号語の有する誤りがシンドロームの半数より少ない場合、このアルゴリズムは成功する。１つの変形例において、標準リードソロモン誤り訂正アルゴリズムが実現されて拡大訂正を実行する。いくつかの状況においては、拡大訂正アルゴリズムはエンティティを訂正することができないが、消失訂正アルゴリズムが仕上げることは可能である。この場合、拡大訂正は、装置が別の反復を試みるべきであることを示す結果を返す。さらに他の場合において、拡大訂正が別の反復を直接試みることができないことは明らかである。これらの場合、全数訂正プロセスが実行される。拡大訂正プロセス１４０全体を示すダイヤグラムは図１４において示される。

【００３０】

全数訂正アルゴリズムは、全数的に訂正組合せを試みることによってエンティティを訂正することを試み、次にＣＲＣ６４検査を用いて訂正を確認する。全数訂正アルゴリズム１５０のフローチャートは図１５に示される。このプロセスは、図１１の「全数訂正を試みる」１１２とラベルをつけられたボックスに対応する。

【００３１】

第１のボックス「全数訂正のための設定」１５２は、全数訂正の試行を始める前に必要な、あらゆる１回限りの計算のためである。これらの計算は主に機能強化のためである。最後のボックスは、他のアルゴリズムがいずれかのＥＣＣブロックを訂正したか否かを見る最終検査である。

【００３２】

全体として、プロセスは特定のアルゴリズムを試み、なんらかの良いＣＲＣ結果があるかを検査するものである。良いＣＲＣ検査がある場合、次に反復的な訂正（十分な良いＣＲＣ検査がある場合）、または、良いＣＲＣ検査をさらに生成しようとして初めにアルゴ

リズムを再開する。アルゴリズムのいずれもが良いＣＲＣ検査を生成することができない場合、全数訂正アルゴリズムは失敗する。失敗の後、装置は再位置決めの再試行を実行するか、または読取り困難な誤り（ＨＲＥ）を報告する必要がある。

【００３３】

図１５の第２のボックス１５４は、いずれかの信頼できる符号語の訂正を試みるアルゴリズムのためのボックスである。全体として、このアルゴリズムは、ＥＤＣを用いて、ＥＤＣ衝突のない符号語を訂正する。このアルゴリズム１６０は、図１６に概説されるステップを含む。このアルゴリズムの第１のステップ１６２は、信頼できない符号語のリストを点検し、いずれかの訂正されたブロックのＣＲＣを装置がうまく確認した場合には、これらを良いとしてマークすることになる。このステップは、有効なブロックを生じた訂正のいずれかの部分に見える場合、信頼できない符号語を信頼できる符号語に発展させている。この符号語が正しくないかもしれないという可能性は依然としてあるが、その確率は、目標とする訂正不能な誤り率よりも、十分低い。前のステップにおいて信頼できない符号語を発展させた後、装置は次に各ブロックのＥＤＣ残余を用いていずれかの信頼できる領域を訂正する。信頼できる領域は、他のいかなる信頼できない、または訂正できない符号語とも衝突を有しない、３２ビット語の一部である。装置はこれらの場合には、ＥＤＣ残余内の不良な符号語と領域との、信頼できる１対１のマッピングを有する。装置は、取消し（undo）情報を保つことについて懸念することなく、これらの変更を適用することができる。

【００３４】

このアルゴリズムは各符号語を調べ、それが衝突を有しないか、部分的な衝突を有するか、完全な衝突を有するかを判断する。衝突がない場合、装置は完全にこの符号語を訂正し、良いとしてマークすることができる。部分的な衝突がある場合、装置は衝突のない領域を訂正することができるが他のビットはそのまま放置する。装置はこの符号語を訂正不能な符号語としておいておく。完全な衝突がある場合、装置は符号語を完全にスキップする。このアルゴリズムでは、装置が衝突を有しない非ゼロのＥＤＣビットを見つけた場合には、いかなる信頼できない符号語も訂正不能な符号語になる。この場合、装置はこれが誤訂正であると仮定して、以前の訂正を取消す。

【００３５】

符号語の信頼できる部分の処理が、成功した訂正をいかに生成するかを示す例が下記に説明される。この例は既存のテープドライブフォーマット（すなわちクウォンタムテープドライブのチポトレ（Ｃｈｉｐｏｔｌｅ）（ＳＤＬＴ３２０）フォーマット）に書込まれたデータの処理を示す。図１７に示すように、例としてのエンティティ１７０は、２つの消失した（完全に失われた）ブロック９および１０を有する。ブロック１から８は、少なくとも１つの破損したバイトに起因する不良なＣＲＣを有する。他の１０のブロックは全て良いＣＲＣを有し、シンドローム生成を除いては訂正に参加しない。

【００３６】

この例では、拡大訂正アルゴリズムは７個の破損した符号語を特定し、上記の図でＡからＦまでラベル付けをした。符号語Ｂ、Ｃ、Ｄ、ＥおよびＧは全て１つの破損しか有さず、全て正しく訂正される。他の２つの符号語（ＡおよびＦ）は、ＥＣＣ２の訂正能力を超える。２つの消失は各々１つのシンドロームを燃焼させるので、誤り訂正に利用可能なのは元の４つのうち２つのシンドロームのみである。誤り訂正は２つのシンドロームを必要とするので、アルゴリズムの訂正の限界は１つの符号語につき１つの誤りとなる。符号語ＡおよびＦは、両方とも２つの誤りを有する。この説明の目的のため、拡大訂正アルゴリズムが符号語Ａを訂正不能として特定したにもかかわらず、誤って訂正符号語Ｆを訂正したと仮定する。結果として生じるレイアウト１８０が図１８に示される。

【００３７】

消失したブロック９および１０はほぼ再生することに注意されたい。符号語が正しい場合、ブロック９および１０の対応する領域もまた正しい。同様に、符号語が訂正不能であるかまたは誤って訂正された場合、９および１０の対応する領域はこれを反映する。

【００３８】
拡大訂正アルゴリズムは、これら全ての符号語の位置をテーブルに記録する。符号語Ａは訂正不能な符号語リストに入り、符号語ＢからＧは信頼できない符号語リストに入る。これらはすべて信頼できない。なぜならば、符号語Ｆの場合と同様、比較的高い確率で不良な訂正があるからである。実際この状況では、およそ１６分の１の確率で誤訂正がある。

【００３９】
拡大訂正アルゴリズムは、完了後、全ての不良なブロックのＥＤＣを検査する。前の図の右列は、不良なＥＤＣ残余にＸをつけて示し、良い残余に空の長方形をつけて示す。したがって、ブロック４から８のすべてが良いＥＤＣ残余を有し、他のブロックは不良な残余を有する。この説明の目的のために、ブロック４から８において続くＣＲＣ検査も良い残余（全零）という結果になると仮定できる。ＣＦ１フィールドまたはＣＲＣ検査バイト自体において問題が生じない限り、たいていこれが当てはまる。

【００４０】
この時点で、拡大訂正アルゴリズムは、５個のブロック（１、２、３、９、１０）が不良なＥＤＣ／ＣＲＣ検査を有することを発見している。これは、反復訂正アルゴリズムによって許容される４個のブロックの限界を超える。これにより全数訂正アルゴリズムに制御を転送する必要が生じる。全数訂正の第１のステップは、ＥＤＣ訂正を試みることである。

【００４１】
このアルゴリズムは、全ての信頼できない符号語（この場合は符号語ＢからＧである）を点検して、訂正されたバイトのいずれかが今や良いＣＲＣ検査を有するブロックと一致するか否かを検査する。信頼できない符号語テーブル１９０が図１９に示される。

【００４２】
ブロック４から８の全てが良いＣＲＣ検査を有するので、これらの符号語のほとんどを信頼できないものから信頼できるものへと発展させることができる。事実、信頼できないままで残る唯一の符号語は符号語Ｆのみである。信頼できない符号語を確認した後のエンティティの状態を示すダイヤグラム２００が図２０に示される。符号語Ａ２０２は訂正不能であり、図２０に示されるように、符号語Ｆ２０４は信頼できない。

【００４３】
これらの２つの符号語がＥＤＣ残余の異なる部分に影響を及ぼす（すなわちそれらがＥＤＣマップにおいて衝突を有しない）確率は約５３％（１７／３２）である。この例については、これらの符号語が独立していると仮定することができる。符号語Ａがオフセット２にあり、符号語Ｆはオフセット１００にあると仮定する。結果として生じる符号語のＥＤＣマップ２１０が図２１に示される。結果として生じる衝突マップ２１２がゼロであることに注意されたい。これは、２つの符号語がＥＤＣ残余の独立領域に影響を及ぼすことを示す。次に２つの符号語に各ブロックのＥＤＣ残余を安心して適用することができる。図２２において、サンプルとしていくつかの誤りパターンが与えられ、これらのパターンがいかにＥＤＣ残余２２０に影響を及ぼすかを示す。この例の全ての自然な誤りは誤りパターン「１０１」を有し、他の誤りはランダムである。

【００４４】
ＥＤＣ残余内のどのビットがどの符号語に対応するかが特定されたので、その後の訂正はこれらの情報の関連付けの問題となる。アルゴリズムの１つの変形例は下記に与えられる：

1　全ての破損したブロックを通ってループする；
2　ブロック内の全ての破損した符号語を通ってループする；
3　現在のブロックおよび符号語については、ＥＤＣ残余とＥＤＣマップとの間の論理積として訂正語を計算する；
4　訂正語と一致するビットを取除くことによってＥＤＣ残余を更新する；
5　訂正語をもとの符号語の位置にローテーションさせる；

６　キャッシュ内で訂正語と破損した語との間で排他的論理和演算を用いてリードーモディファイーライトを実行する；
７　他の全ての符号語およびブロックを続ける。
例として、図２２のブロック１内の破損および符号語Ａへのアルゴリズム２３０の適用が図２３に示される。

【００４５】

最後のステップは、キャッシュに排他的論理和演算を実行することによってキャッシュ内のデータに最終訂正語を適用することである。全てのゼロビットはキャッシュデータを不変に保ち、１ビットがキャッシュデータを反転させる。全てのＥＤＣ訂正を実行した後、最終ステップは、今や良いＥＤＣ残余を有する（すなわち全零）いずれかのブロックのＣＲＣを検査することである。上記の例は、ＥＤＣ訂正アルゴリズムが、ＥＤＣ残余からの情報を用いて、独立符号語を含むエンティティをいかに訂正することができるかを示す。次のセクションでは、従属符号語（すなわち非ゼロの衝突マップ）を有する場合をいかに扱うかを説明する。

【００４６】

図２４に示すように、あるアルゴリズムが与えられて、恐らく誤訂正であろう符号語を見つけ、次に訂正を取消す。しばしばこのステップは、良いＣＲＣを有するブロックを生じる。このアルゴリズムは、信頼できない訂正が誤訂正であると仮定して、信頼できない符号語を訂正不能な符号語に変換することを試みる。その後、フローは次のアルゴリズム（すなわち訂正不能な符号語の試行）へと続き、その時点で、訂正不能な符号語を信頼できない符号語へ再び変換する試みが行われる。これらのアルゴリズムを２回以上試みるのが必要な状況があるので、図１５のマスターフローはこれらの２つのアルゴリズムを２度含む。

【００４７】

図２４を参照すると、信頼できない符号語を試行するためのアルゴリズムが詳細に示される。このアルゴリズムの機能の１つは、誤訂正のための最良の候補を特定することである。１つの変形例において、１ブロックについて約５個から２０個の信頼できないオフセットがある。大部分の訂正は、１ビットまたは２ビットのいずれかである。これは自然に発生する誤りが典型的には１ビットのみであるという事実に起因し、アンプリコーダ（un-precoder）が、それをパターン１０１まで拡大する。欠陥がバイトの終わりまたは始まりに近い場合、第２のビットは１ビットの誤りを残してバイトの外側で拡大する。

【００４８】

他方、誤訂正は、典型的にはランダムな８ビットパターンである。これは、８ビットパターンのいずれかが有効な符号語を作る確率が均一に分散されているからである。有限体は合理的な疑似乱数を生成することができ、この場合、誤訂正は疑似乱数に類似する。

【００４９】

このアルゴリズムは、最大数の２進法の１を含む、ＥＤＣ残余内でパターンと一致する訂正を検索することにより、最も信頼できない符号語を見つける。図２５は、一様な乱数を仮定して、ある数のビットを得る確率２５０を示す。図２５に示すように、乱数は３、４または５ビットを支持する傾向があるのに対し、自然な誤りは１または２ビットを支持する傾向がある。より大きいビット数を有する訂正を選択することによって、良い訂正よりも誤訂正を選択する確率が極めて高くなる。

【００５０】

次のセクションは第４のボックス、図１５の訂正不能符号語の試行を説明する。このアルゴリズムは、信頼できない訂正を除き、以前には訂正不能であったオフセットのみに焦点を当てること以外はＥＤＣ訂正アルゴリズムに類似する。このエンティティに誤訂正がない（すなわち全ての「信頼できない」訂正が信頼できる）場合、このアルゴリズムはしばしば成功する。このアルゴリズムに問題を起し得る他の条件は、２つの訂正不能な符号語が全てのデータブロックにおいて衝突する破損を有する場合である。この状態では、ＥＤＣは２つ以上の符号語の混合体を有し、いかなる特定の符号語からも正しい値を分離す

ることが不可能となる。

【００５１】

図２６は、全ての（以前には）訂正不能な符号語を訂正するよう試みるためのアルゴリズム２６０を示す。第１のステップ２６２は、訂正不能な符号語のための衝突マップを計算する。このマップは全てのエンティティについて一定であって、各符号語の性質を特定するためのガイドを与える。次にアルゴリズムは、不良なＥＤＣを有する全てのデータブロックを通ってループする。各ブロックについて、装置は、１）衝突のない符号語を扱う、２）衝突を有する符号語を扱う、という２つのアルゴリズムの各々を試みる。これら２つのアルゴリズムのいずれかが良いＣＲＣを有するブロックを見つけると、図２６のアルゴリズムは成功２６４に戻る。図１５のアルゴリズムは次に、装置が反復的訂正のために十分なマージンを有するか否かを検査し、装置が十分なマージンを有しない場合には最初から続ける。

【００５２】

次の２つのセクションは、図２６の２つの副アルゴリズムを説明する。第１のセクションは、図２６の「衝突のない符号語の訂正」２６６とラベルをつけられたボックスを説明する。衝突がないので、このアルゴリズムは多くても４つの処理すべき符号語を有する。この制限が４であるのは、３２ビットのＥＤＣ残余に、４バイトを超えて独立に適合させることは不可能だからである。衝突を有する符号語と別個にこれらの符号語を処理することが効率的であるのは、これらの符号語がブロック内で値を変えないからである。逆に、衝突を有する符号語はいくつかのあり得る値を有する。

【００５３】

このアルゴリズムの第１のステップは、衝突マップ（Ｃｏｌｌｉｓｉｏｎ　Ｍａｐ）、完成マップ（Ｃｏｍｐｌｅｔｅ　Ｍａｐ）、およびＥＤＣ残余（ＥＤＣＲｅｓｉｄｕａｌ）を用いて、このアルゴリズムが所望の結果を生成することができるかどうかを判断することである。プロセスの１つの変形例が以下に詳細に記載される：

１．ＥＤＣ残余がゼロであり、ＣＲＣ検査が失敗した場合、装置はこのブロックを処理することができない。次のブロックへ進む；

２．ＥＤＣ　Ｒｅｓｉｄｕａｌ　ＡＮＤ　ＮＯＴ　Ｃｏｍｐｌｅｔｅ　Ｍａｐを計算する（すなわち、Ｃ表記でＥＤＣ＆～ＣｏｍｐｌｅｔｅＭａｐ）。この結果が非ゼロである場合、訂正不能な符号語のいずれもが影響を及ぼすことができないビットがＥＤＣ残余内にある。この場合、装置はこのブロック全体をスキップする。符号語のうち１つに誤訂正があった；

３．前の検査が成功した場合、装置は次に、衝突のない符号語に対応する非ゼロのビットがＥＤＣ残余内にあるか否かを見る。装置は、「ＥＤＣ　Ｒｅｓｉｄｕａｌ　ＡＮＤ　ＮＯＴ　Ｃｏｌｌｉｓｉｏｎ　Ｍａｐ」（すなわちＣにおけるＥＤＣ＆～ＣｏｌｌｉｓｉｏｎＭａｐ）としてこれを計算することができる。この結果がゼロである場合、装置は次に衝突を有する訂正不能な符号語を扱うためのアルゴリズムにスキップする。そうでなければ、装置はこのアルゴリズムを続ける。

【００５４】

前の検査が成功した場合、装置は次に衝突マップ外でＥＤＣ残余の一部に影響を及ぼす訂正不能な符号語について検索する。装置がこのような符号語を見つけた場合、装置は衝突マップと一致しない符号語の一部を訂正する。この符号語は、誤り値としてＥＤＣ残余を用いる。

【００５５】

第２のセクションは、衝突を有する訂正不能な符号語の組合せを全数的に選択することによって訂正を試みるアルゴリズム２７０を説明する。図２７は、アルゴリズムを概説する。

【００５６】

一般的な概念は、このアルゴリズムを用いて従属的な訂正不能な符号語の全ての組合せを試みることである。しかしながら、このアルゴリズムが盲目的に全ての組合せを試みる

とすると、極めて多数の可能性があり得ることとなる。特に３つの消失があるフォーマットの場合にそうである。この場合、全ての破損が訂正不能であり、潜在的に多数の組合せがある。推定される組合せの最大値は、16^4＝６４Ｋである。装置が各データブロックについてこれを行う場合、１００万の異なる可能性があり得る。これには数秒または数分もかかり、装置の再試行はその間によりうまく行われる。

【００５７】

あらゆる組合せを試みる代わりに、このアルゴリズムはそれ自体を合理的な可能性の領域に限定する。第１のアルゴリズムは、あらゆる組合せを試みる前に、まず各符号語に注目し、シンドロームを用いて結果をクロスチェックする。訂正試行が合理的に見えた場合、装置は符号語を有用とマークする。このクロスチェックは、間違った選択肢のうち少なくとも９０％、恐らく９９％を除去する。

【００５８】

シンドロームを修正するための式は次の通りである。

$S'_j = S_j + X^j Y \sigma(X)$

この式において、S'_jは第ｊの新しいシンドローム、S_jは現在のシンドローム、Ｘは現在のブロックの位置、ＹはＥＤＣの誤りパターンであって、σ（Ｘ）はＸにおいて評価された消失位置多項式である。変数ｊは０から（ＥＣＣＢｌｏｃｋｓ－Ｅｒａｓｕｒｅｓ－１）である。

【００５９】

良いクロスチェックを用いて全ての符号語を見つけた後、装置はこれらの符号語がＥＤＣ残余の非ゼロ領域の完全なカバレージを有するか否かを再確認することができる。装置は、これらの符号語について全てのＥＤＣマップの論理和を計算し、次にＥＤＣ残余を用いて論理積演算を実行することによって、このカバレージを決定する。この結果が非ゼロである場合、装置はこのブロックを訂正するのに充分なカバレージを有しない。この場合、装置は次のブロックにスキップする。

【００６０】

装置が完全なカバレージを有する場合、次のステップは、全零のＥＤＣ残余を残す独立した符号語の全ての組合せを試みることである。ＥＤＣ残余がゼロになる場合、装置はこのブロックを最終的に確認するためにＣＲＣ検査を試みることができる。単純な再帰的アルゴリズムは、符号語を確認しながら前の符号語から独立している新しい符号語を選択することによって、これらの組合せを見つけることができる。

【００６１】

下記のセクションでは、３２ビットのＥＤＣ（誤り検出符号）およびこの符号がエンティティ訂正アルゴリズムにいかに役立つことができるかを説明する。ＥＤＣは、データブロック内のユーザデータ領域を保護する、３２ビットの誤り検出符号である。ＥＣＣブロックは有効なＥＤＣフィールドを含まない点に注意されたい。

【００６２】

下記のセクションでは、ＥＤＣ、ＥＤＣマップの特性を説明し、次にＥＤＣ訂正アルゴリズムに及ぶ。要するに、ＥＤＣマップは、破損した符号語をＥＤＣ残余の対応するセクションに接続する方法である。それはＥＤＣおよび全数訂正アルゴリズムにおいて用いられる。計算されたＥＤＣ残余は誤りパターンのみに依存し、良いデータには依存しない。さらに、ＥＤＣは線形である。

【００６３】

ＥＤＣ計算は、リードソロモン符号またはＣＲＣ検査（または他のＢＣＨ符号）によって用いられる計算と同様の、ガロア体の計算を用いる。

【００６４】

ＥＤＣは３２ビット記号を用い、ＧＦ（2^{32}）のリードソロモン符号（３２ビットのガロア体）と同様である。このガロア体のための定義多項式Ｐ(ｘ)は次の通りである。

【００６５】

$P(x) = x^{32} + 1 = (x+1)^{32}$

ＧＦ（2^{32}）における原始元αは、
α＝（00000000.00000000.00000000.00000010）（２進法）
である。
【００６６】
標準リードソロモン符号とは異なり、この符号はその計算に原始多項式を用いない。事実、ガロア体のサイズは、完全な$2^{32}-1$ではなく、３２元のみである。
【００６７】
この原始多項式は、ハードウェアおよびファームウェアにおいて非常に使いやすい。なぜならばαによる乗算は、１ビットの左ローテーションと等価だからである。事実、α^{N}を乗算することはＮビットの左ローテーション（ｍｏｄ３２）と同じである。
【００６８】
ＥＤＣは、次のような生成多項式Ｇ(ｘ)を有する。
Ｇ(ｘ)＝ｘ＋α
この例では、生成多項式は非常に単純な形式を取る。しかしながら、別のアルゴリズムが誤り位置を決定することができる場合、ＥＤＣは単一の３２ビット記号を訂正することができる。これは全数訂正アルゴリズムにおいて用いられる重要な機能である。図２８は、ＥＤＣジェネレータ／チェッカ回路２８０の表示である。
【００６９】
上記のダイヤグラムにおいて、プラスのついた円はガロア体加算を示し、それはビットワイズＸＯＲ（排他論理和）演算の等価物である。αのついた円はガロア体乗算演算であって、この場合１ビットの左ローテーション演算と同じである。「ＥＤＣ」とラベルのついたボックスは、ＥＤＣ残余を保持する３２ビットレジスタである。ＥＤＣ計算を始める前に、ハードウェア（またはファームウェア）は「ＥＤＣ」レジスタをすべて１に初期化する。初期化後、ハードウェアは全てのデータを順に回路に入れる。ハードウェアは、各クロックサイクルにおいて、データの１つの３２ビット語を処理する。
【００７０】
ＥＤＣ演算は、リードソロモン符号と同じく線形である。この特性は、それにより良いデータを無視して誤りパターンのみに焦点を当てることができるので、有用である。ここでこれがいかに作用するかを説明する。
【００７１】
ｍ（ｘ）がメッセージまたはオリジナルデータであるとする。ｅ（ｘ）が誤りパターンである。これらの各々が多項式であって、そこでデータまたは誤りパターンは、多項式内の各項の係数である３２ビット語である。最初の３２ビット語はｘの最大冪乗の係数であり、最後の３２ビット語は定数項（すなわちx^{0}の係数）である。
【００７２】
受取ったデータｒ(ｘ)は、メッセージを誤りパターンに加算した結果である。
ｒ（ｘ）＝ｍ（ｘ）＋ｅ（ｘ）
ＥＤＣ（ｆ（ｘ））はデータストリームｆ（ｘ）上でＥＤＣ残余を計算する演算であるとする。ＥＤＣ計算は、下記の式が有する性質を示す。
【００７３】
ＥＤＣ（ｆ（ｘ））＝ｆ（ｘ）ｍｏｄ（ｘ＋α）＝ｆ（α）
これは、ＥＤＣ残余が、ｘ＝αにおける入力多項式の評価と同じであることを意味する。データが良い場合、ＥＤＣ残余はゼロである。すなわち、
ＥＤＣ（ｍ（ｘ））＝ｍ（α）＝０
となる。
【００７４】
装置が受取ったデータのＥＤＣを次に計算する場合、装置は以下を得る。
ＥＤＣ（ｒ（ｘ））＝ｒ（α）＝ｍ（α）＋ｅ（α）
ｍ（α）＝０をＥＤＣ（ｒ（ｘ））＝ｍ（α）＋ｅ（α）に代入すると、装置は、
ＥＤＣ（ｒ（ｘ））＝ｅ（α）

を得る。

【００７５】

これが意味することは、装置が良いデータを無視して誤りパターンにのみ焦点を当てることができるということである。すなわち、ＥＤＣ残余内のいかなる１も誤りパターンの直接の結果であって、良いデータから独立している。

【００７６】

この例では、ＥＤＣ計算は、データがＥＤＣ計算の前にスワップされるか、または後にされるかに依存しない。ポリシープロセッサは、スワップされた順にキャッシュからデータを読取る。この文脈においてスワップとは、３２ビットデータ語内で、高位の１６ビットをより低位の１６ビットに交換することを指す。これは、ＥＤＣ結果の単一のスワップとは反対に、全データ語におけるスワップを避けることによって計算が省力化される。この特性を要約する式は以下の通りである。

【００７７】

ＥＤＣ（ＳＷＡＰ（Ｄａｔａ））＝ＳＷＡＰ（ＥＤＣ（Ｄａｔａ））

これが当てはまるのは、１６ビットのスワップ演算が１６ビットの左ローテーションと等価であり、ガロア体の計算が線形だからである。ＥＤＣのガロア体において、スワップはα^{16}による乗算に等しい。ガロア体の計算においては、ユーザデータを、各データ点を各項の係数として多項式として表示するのが便利である。ここで、Ｎ個の３２ビットのデータ語があると仮定して、データストリームｍ（ｘ）を表示する式がある：

【００７８】

【数１】

$$m(x) = D_0 x^{N-1} + D_1 x^{N-2} + \ldots + D_{N-2} x + D_{N-1}$$

【００７９】

ここで、$D_{0..N-1}$はユーザデータストリームである。

装置が全てのデータをスワップする場合、メッセージデータは以下のようになる。

【００８０】

【数２】

$$m_{swap}(x) = \alpha^{16} D_0 x^{N-1} + \alpha^{16} D_1 x^{N-2} + \ldots + \alpha^{16} D_{N-2} x + \alpha^{16} D_{N-1}$$

【００８１】

ガロア体の計算は分配法則に従い、それはａｂ＋ａｃ＝ａ(ｂ＋ｃ)とする。この法則を用いて、装置は次のようにα^{16}因のすべての因数を集めることができる。

【００８２】

【数３】

$$m_{swap}(x) = \alpha^{16}(D_0 x^{N-1} + D_1 x^{N-2} + \ldots + D_{N-2} x + D_{N-1}) = \alpha^{16} m(x)$$

【００８３】

データストリームのＥＤＣ残余は、ｍ（ｘ）を（ｘ＋α）で除算した後の剰余と等価である。これは、評価ｍ（α）と等価である。スワップされたデータストリームのＥＤＣを得た結果は次の通りである。

【００８４】

【数４】

$$EDC(m_{swap}(x)) = m_{swap}(\alpha) = \alpha^{16} m(\alpha) = SWAP(EDC(m(x)))$$

【００８５】

要約すると、ＥＤＣ残余をスワップする限り、データストリームの各語をスワップする必要はない。また、ポリシープロセッサにとってはスワップされた残余がより便利なので、下記に記載されるＥＤＣマップは全てスワップされたバージョンのＥＤＣ残余を用いることにも注意されたい。これが適用されるのは、全てのマップが一貫したスワップされる

順序を有するからである。

【００８６】

ＥＤＣマップ（ＥＤＣマスクとも呼ばれる）は、特定の符号語とＥＤＣ残余の対応する領域との関連をマッピングするために用いられる構成である。このマッピングは、ＥＤＣ残余が信頼できるか否かを判断するのを助け、全数訂正アルゴリズムのために有用である。誤りがいかにＥＤＣ残余に影響を及ぼすかを示す例は以下に記載される。まず、ブロック内で１バイトの誤りを有するＥＤＣ残余がいかなるものかを示す例が下記に説明される。

【００８７】

第１０の３２ビット語に以下の誤りパターンＥがあるとすると、

Ｅ＝00000000.00000000.00000000.11100111

である。

【００８８】

この誤りパターンは、最初の３バイトには良いデータ、最後のバイトには破損データを表示する。ＥＤＣが線形であるので、装置は良いデータを完全に無視することができ、誤りパターンだけに焦点を当てることができる。全ての良いデータはゼロに等しいと仮定することができる。たとえば、１００個の３２ビット語がＥＤＣ保護された領域内にあると仮定すると、結果として生じるＥＤＣ残余を決定するために、装置は、全零のシードだけを用いて、単一の誤りパターン上でのみＥＤＣを計算することを想定し得る。この例では、シードが全零であり、データが全零なので、最初の９個の演算は何もしない。装置が第１０の語を処理するときに、ＥＤＣは誤りパターンＥを含む。この時点以降、ＥＤＣ計算は全零の語の処理を続ける。全零の語はＥＤＣに影響を及ぼさないので、唯一生じるのは、誤りパターンがあらゆる新しい語について１ビットの左ローテーションを続けることである。装置が最終的に最後の語に達するとき、装置はＥＤＣ値において１００－１０＝９０のローテーションを実行している。３２ビットの巡回ローテーションが元の値を与えるので、装置は、ローテーション数ｍｏｄ３２（すなわち３２で乗算した剰余）のみを見ればよい。

【００８９】

ローテーション＝９０ｍｏｄ３２＝２６

したがって、結果としてのＥＤＣ残余は、２６ビットだけ左ローテーションした誤りパターンＥである。以下が結果である。

【００９０】

ＥＤＣ＝10011100.00000000.00000000.00000011

装置が別の手段（例えば拡大訂正からの失敗した符号語）によって、破損したバイトの位置を特定することが可能な場合、装置は、ＥＤＣ残余を２６ビットだけ右ローテーションをさせ、次にこの値を第１０の３２ビット語で排他的論理和をとることにより訂正を実行することができる。これがＥＤＣ訂正アルゴリズムの奥にある概念である。

【００９１】

エンティティ内の各バイト幅の符号語は、ＥＤＣ残余内のビットの特定の組を変える能力しか有さない。これが当てはまるのは、循環ローテーション演算が３２ビット語にわたってパターンを広げないからである。その代わり、循環ローテーションは元のビットを合わせて維持し、単にその位置を変える。この特性のために、符号語によって恐らく影響を受けるであろう３２ビット語の領域、および符号語の相対的なエンティティオフセットを決定しさえすればよい。

【００９２】

上記の例において、装置は第１０の語の最終バイトが何らかの破損を有することを知っている。しかし装置は、誤りパターンが何であるかについてはまだわからない。装置は、誤りが最終バイト内でビットに影響を及ぼすが、語の他の３つのバイトには影響を及ぼさないことを特定しただけである。装置はこの情報を用いて下記のパターンを有するＥＤＣマスクを生成することができる。

【００９３】
Ｍａｓｋ＝00000000.00000000.00000000.11111111
このマスクは、ビットが誤りパターンに依存し得ることを示すために２進法の「１」を用い、ビットが誤りパターンから独立していることを示すために２進法の「０」を用いる。この規則によって、正しいローテーションを用いて、全てのＥＤＣマスク間で論理和演算を実行することによって、影響を受け得る全てのビットを見つけることが可能である。
【００９４】
ＥＤＣマップアルゴリズムは、全てのＥＤＣマスクおよび残余を中立位置までローテーションさせ、これは定義上、オフセットゼロのためのローテーションである。この例において、このマスクが語１０にあるので、装置はマスクを１０ビットだけ右ローテーションさせることによってマスクを「正規化」することが必要である。正規化されたマスクは以下の通りである。
【００９５】
Ｍａｓｋ＝00111111.11000000.00000000.00000000
このマスクをＥＤＣ残余と比較するために、装置はＥＤＣ残余を１００ビットだけ右ローテーションさせることによって正規化する必要がある。なぜならば以前の例で、ＥＤＣを語１００に配置したからである。第３２、第６４、および第９６のローテーションは全て残余をもとの位置に戻し、そのため装置は残余を１００－９６＝４ビットだけ右へ動かせばよいだけである。ここで結果として生じるＥＤＣ残余がどのようになるか、例として、ＥＤＣ＝10011100.00000000.00000000.00000011を用いて示す。
【００９６】
ＥＤＣ＝00111001.11000000.00000000.00000000
ＥＤＣ＝00111001.11000000.00000000.00000000から正規化されたＥＤＣ残余が、Ｍａｓｋ＝00111111.11000000.00000000.00000000からのマスクと一致することに注意されたい。これが、装置がＥＤＣマスク（またはＥＤＣマップ）をＥＤＣ残余に関連づけるやり方である。装置は、以前の右ローテーションと同じ回数だけ左ローテーション演算を実行することによって、これらの正規化されたバージョンをもとのバージョンに戻すよう関連づけることができる。
【００９７】
２つ以上の破損した符号語がＥＤＣ残余内の同じビットに影響を及ぼすことが果たして可能か否かについて判断することは有用である。影響を及ぼせる場合、２つの破損した符号語が互いをキャンセルすることが可能であり、それらは検出されないまま残る。この可能性がある場合、装置はもはやＥＤＣが確実に誤りを検出するとは当てにできず、より強力な誤り検出符号、例えばＣＲＣ６４に切り替えなければならない。衝突もＥＤＣ訂正アルゴリズムに影響を及ぼす。なぜならば、装置は、ＥＤＣ残余の一部を特定の符号語に直接適用する代わりに、訂正試行の複数の組合せを用いなければならないからである。
【００９８】
例えば、エンティティがちょうど拡大訂正を終えても、依然２つの信頼できない（または訂正不能な）符号語を有する。簡潔にするため、フォーマットがレガシテープフォーマット（すなわちクウォンタムＳＤＬＴ６００フォーマット）であり、５個の部分的に回復されたブロックおよび１５個の良いブロックがあると仮定する。符号語はそれぞれバイトオフセット１００および２００にある。図２９は、このシナリオを示すダイヤグラム２９０である。
【００９９】
図２９において、「？」は未知の性質を有するバイトを示す。これらは、訂正不能な符号語の結果であるか、または誤った訂正をされた符号語のあり得る結果である。各符号語内のどのバイトが良く、または不良なのかを判断するのは困難である。エンティティ内の他のバイトは全て良い。
【０１００】
これらの２つの符号語がＥＤＣ残余の異なる部分に対応する場合、これらの符号語の両

方を訂正するためにＥＤＣ残余を用いることが可能である。その上、これらの符号語が独立している場合、装置は破損を特定するためにＥＤＣに依存することができる。これらの符号語両方のためのＥＤＣマップ３００が図３０に示される。

【０１０１】

２つの符号語の各々については、装置は最初のＥＤＣマップ３３２を生成することから始める。このマップは、３２ビット語内の符号語の位置に依存する。それは、バイト０、１、２または３のいずれかに影響を及ぼし得る。この場合、符号語１００および２００は両方ともバイト０に影響を及ぼす（１００ｍｏｄ４＝０、および２００ｍｏｄ４＝０）。例えばＳＤＬＴのビッグエンディアンシステムなどのシステムにおいて、バイト０は最上位の８ビットを占めるので、両方のマップは８個の最上位ビットの１から始める。次に装置はＥＤＣマップを正規化する。これは、最初のＥＤＣマップ３３２をワードオフセットだけ右ローテーションすることによって作用する。符号語１００および２００のワードオフセットは、それぞれ２５および５０である。これは、３２ビット語につき４バイトがあるという事実による（１００／４＝２５および２００／４＝５０）。

【０１０２】

衝突マップ３３６は、全ての正規化されたＥＤＣマップ３３４の中の重なり合う領域である。これは、２つの符号語の場合、２つのマップの論理積である。図３０に示すように、衝突マップ３３６はビット６にただ１つの１を有し、非ゼロである。結果として、装置はＥＤＣ残余のビット６を信頼することができない。それは重要な衝突ではないが十分であるので、装置はブロック確認のためにＣＲＣ検査まで遷移する必要がある。完成マップ３３８は、ＥＤＣ内の全ての影響を受けたビットを示す。これは個々のＥＤＣマップ全ての単純な論理和である。このマップは、一般的な場合に衝突を見つけるとき、３個以上の信頼できない符号語があり得るときに必要である。衝突を見つけるためには、装置は新しい正規化された各ＥＤＣマップを現在の完成マップと比較する。いずれかのビットが重なる場合、装置はそれらを衝突マップに加える。

【０１０３】

本発明が記載され、本発明の具体例が描写された。本発明が特定の変形例および示した図に関して記載される一方、当業者は、本発明が記載された変形例または図面に限定されないことを認識する。加えて、上述の方法およびステップがある順序で発生するある事象を示す場合、当業者はそのステップの順序が変更されてもよく、そのような変更が本発明の変形例に従っていることを認識する。さらに、ステップのいくつかは、可能な場合に並列のプロセスで並行して実行されてもよく、上述のように連続して実行されてもよい。したがって、本発明の変形例があり、請求項に示される本発明の開示または等価物の精神の範囲内で、この特許が同様にそれらの変形例を含むことが意図される。最後に、本願明細書において引用される全ての公報及び特許出願は、個々の刊行物または特許出願が本願明細書において特別かつ個別に出されたかのように、それら全体が本願明細書において引用にて援用される。

【図面の簡単な説明】

【０１０４】

【図１】誤り検出装置が磁気テープドライブ用コントローラの一部である、誤り検出装置の１つの実現例を示す図である。

【図２】テープドライブ用のデータブロックのレイアウトの例を示す図である。

【図３】ＥＤＣを計算するためのアルゴリズムを例示する図である。

【図４】ＥＣＣブロックのレイアウトの例を示す図である。

【図５】ＢＲＣ（後方読取り互換性）フォーマットためのデータおよびＥＣＣブロックのレイアウトを示す図である。

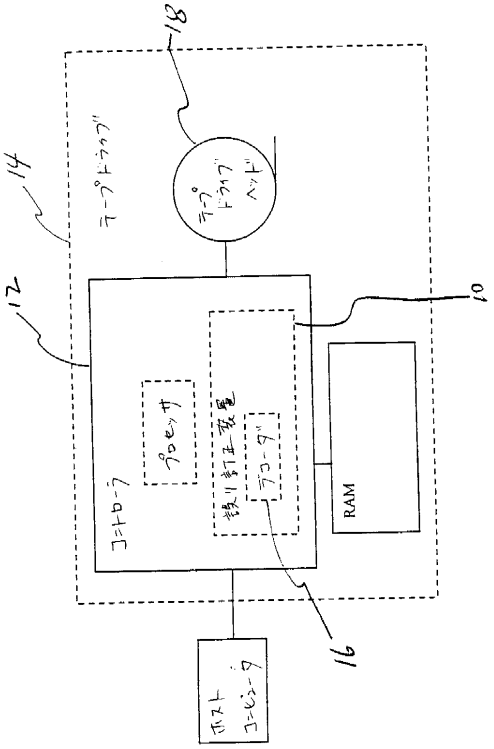
【図６】エンティティのレイアウトの例を示す図である。

【図７】エンベロープのレイアウトの例を示す図である。

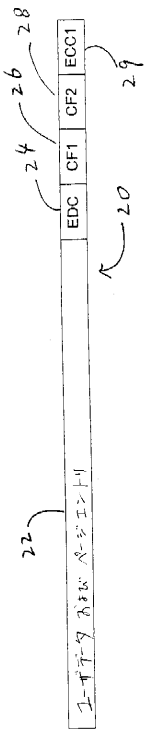
【図８】後方互換性を有するフォーマットの１つのための典型的なエンベロープのレイアウトを示す図である。

【図９】後方互換性を有するフォーマットである別のエンベロープのレイアウトを示す図である。
【図１０】読取りアルゴリズムの例を示す図である。
【図１１】全体的な訂正プロセスを示す図である。
【図１２】訂正タイプの決定のプロセスを示す図ある。
【図１３】消失訂正アルゴリズムの例を示す図である。
【図１４】拡大訂正アルゴリズムの例を示す図である。
【図１５】全数訂正アルゴリズムの例を示す図である。
【図１６】符号語の信頼できる部分を処理するためのアルゴリズムの例を示す図である。
【図１７】ＥＤＣ訂正のためのセットアップの例を示す図である。
【図１８】拡大訂正の結果を示す図である。
【図１９】信頼できない符号語マップを示すテーブルを示す図である。
【図２０】信頼できる符号語および良いブロックを取除いた後のエンティティを示す図である。
【図２１】符号語ＡおよびＦのためのＥＤＣマップを示す図である。
【図２２】ＥＤＣ残余における誤りパターンの効果を示す例を示す図である。
【図２３】最終訂正語の生成の例を示す図である。
【図２４】信頼できない符号語の試用のためのアルゴリズムの例を示す図である。
【図２５】バイト内の所与の１の個数についての確率分布の例を示すダイヤグラムである。
【図２６】訂正不能な符号語の試用のためのアルゴリズムの例を示す図である。
【図２７】衝突を有する訂正不能な符号語の試用のためのアルゴリズムの例を示す図である。
【図２８】ＥＤＣジェネレータ／チェッカ回路の例を示す図である。
【図２９】２つの信頼できない符号語を有するエンティティを示す図である。
【図３０】衝突マップの生成の例を示す図である。
【符号の説明】
【０１０５】
３３２　ＥＤＣマップ、　３３４　正規化されたマップ、　３３６　衝突マップ、３３８　完成マップ。

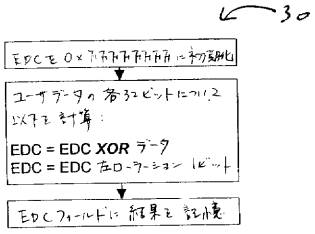
【図１】


18
14
テープドライブ
テープドライブヘッド
12
10
コントローラ
プロセッサ
誤り訂正装置
デコーダ
RAM
16
ホストコンピュータ

【図２】


28
26
24
EDC
CF1
CF2
ECC1
20
29
22
ユーザデータおよびページエントリ

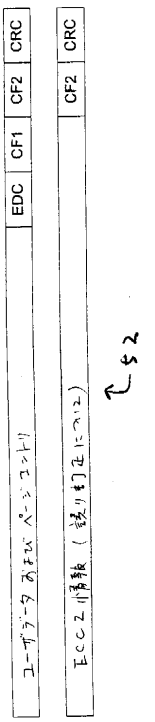
【図３】


30
EDCを0×FFFFFFFFに初期化
ユーザデータの各32ビットについて以下を計算：
EDC = EDC XOR データ
EDC = EDC 左ローテーション1ビット
EDCフィールドに結果を記憶

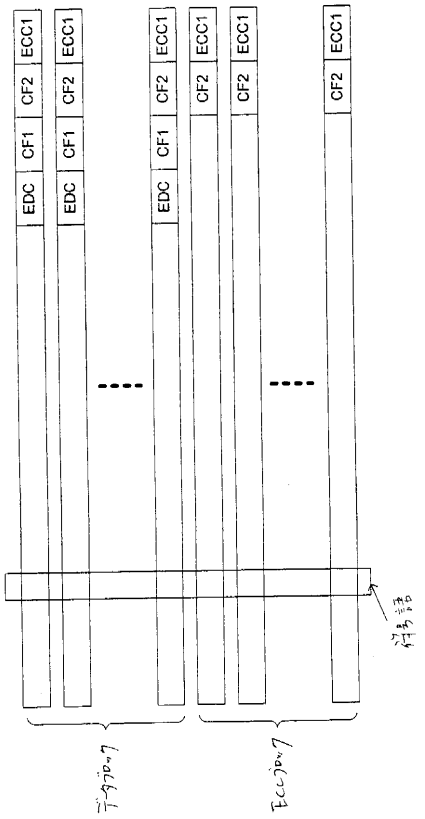
【図４】


44
42
ECC1
CF2
40
ECC2情報（誤り訂正コード）

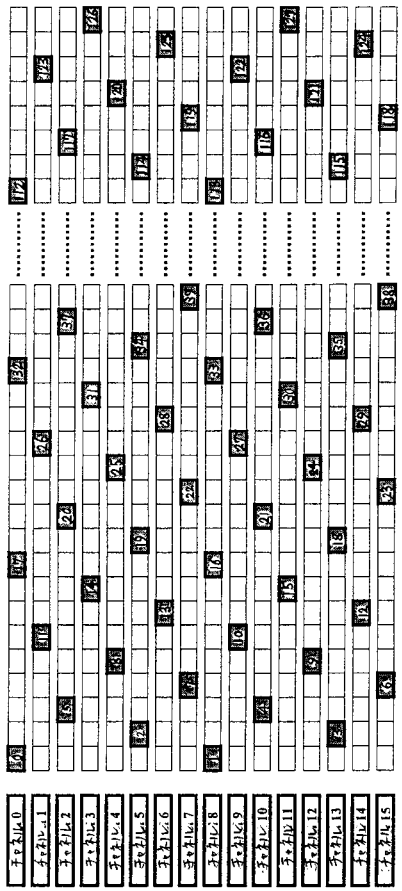
【図５】


EDC
CF1
CF2
CRC
CF2
CRC

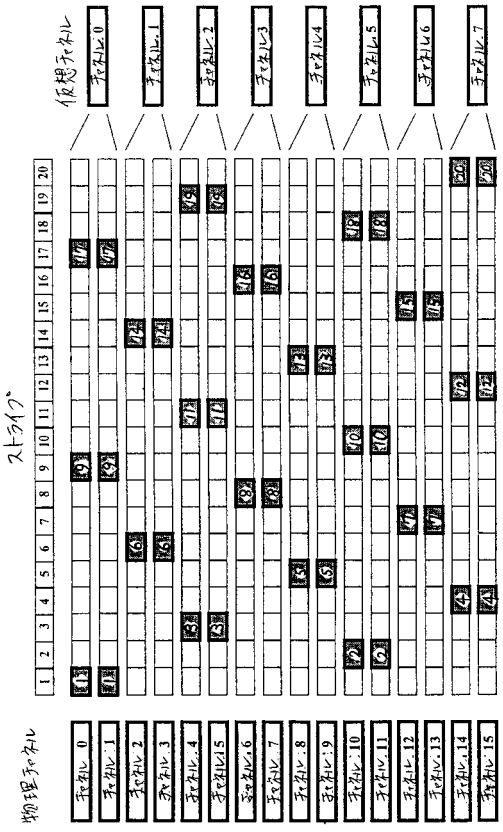
【図６】


EDC
CF1
CF2
ECC1
CF2
ECC1

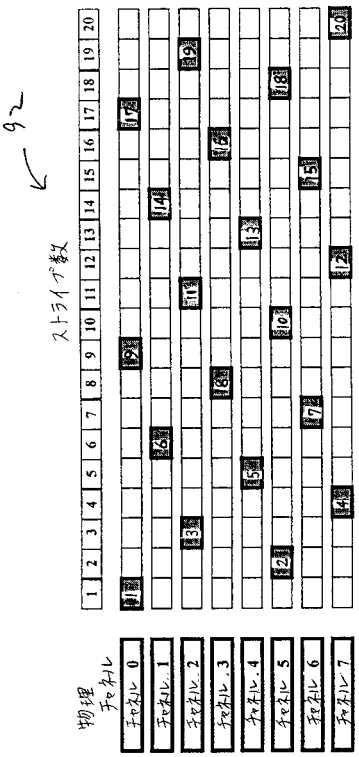
【図７】


ストライプ数
チャネル 0
チャネル 1
チャネル 2
チャネル 3
チャネル 4
チャネル 5
チャネル 6
チャネル 7
チャネル 8
チャネル 9
チャネル 10
チャネル 11
チャネル 12
チャネル 13
チャネル 14
チャネル 15

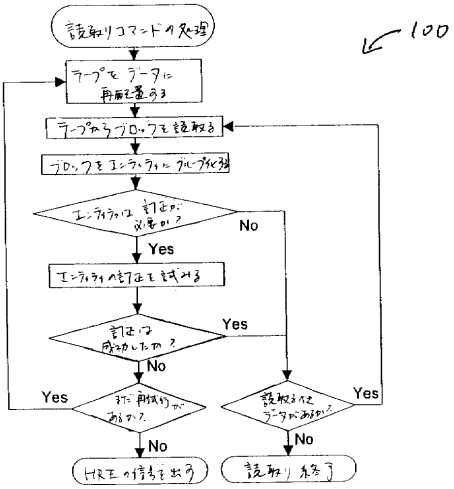
【図８】


ストライプ
仮想チャネル
物理チャネル
チャネル 0
チャネル 1
チャネル 2
チャネル 3
チャネル 4
チャネル 5
チャネル 6
チャネル 7

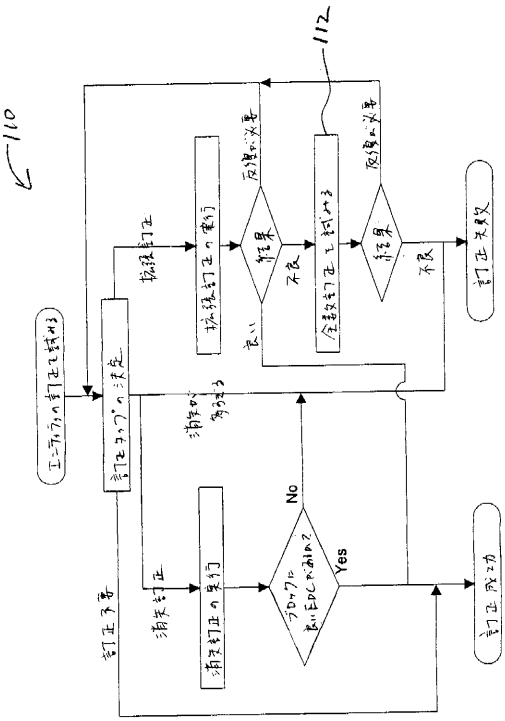
【図９】



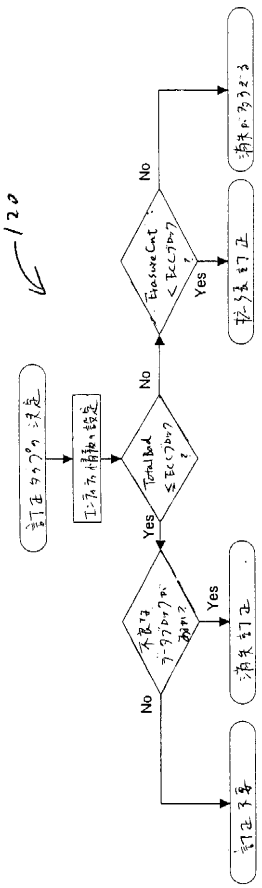
【図１０】



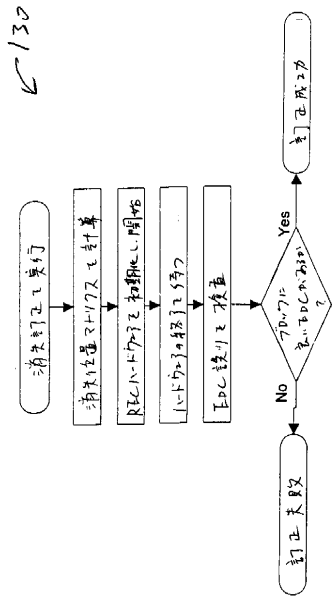
【図１１】



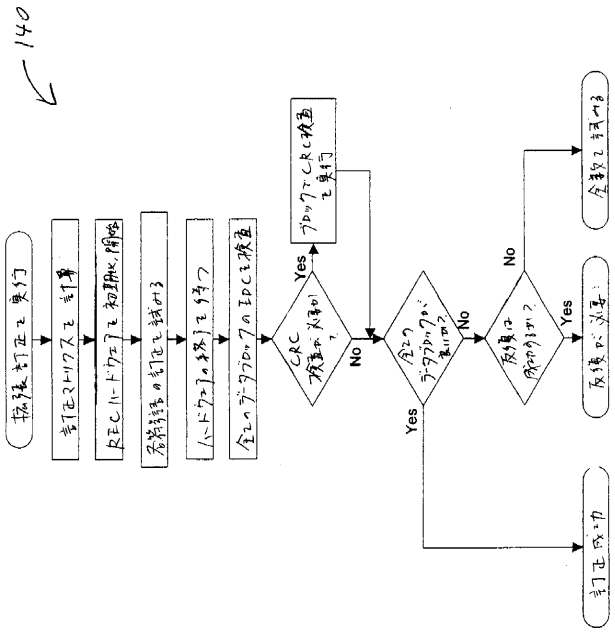
【図１２】



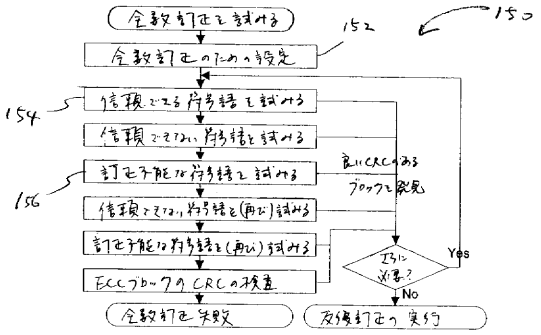
【図１３】



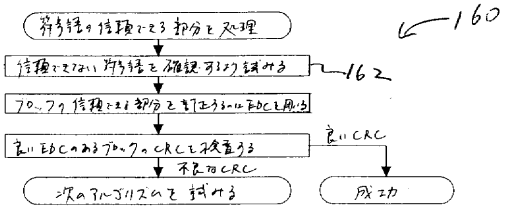
【図１４】



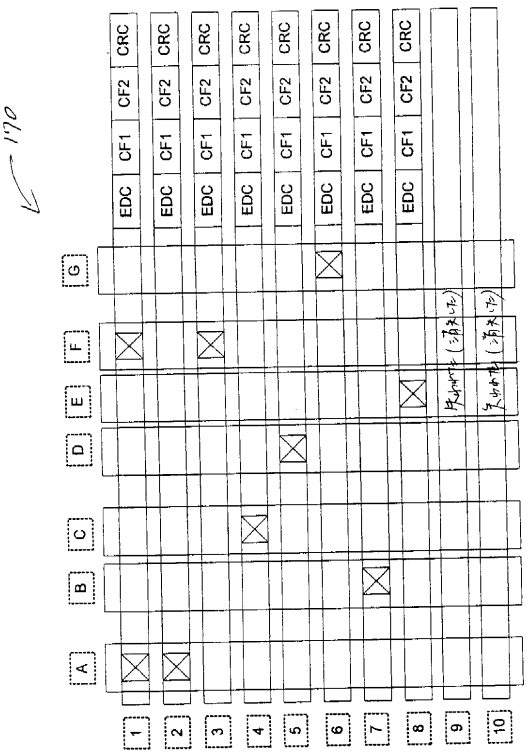
【図１５】



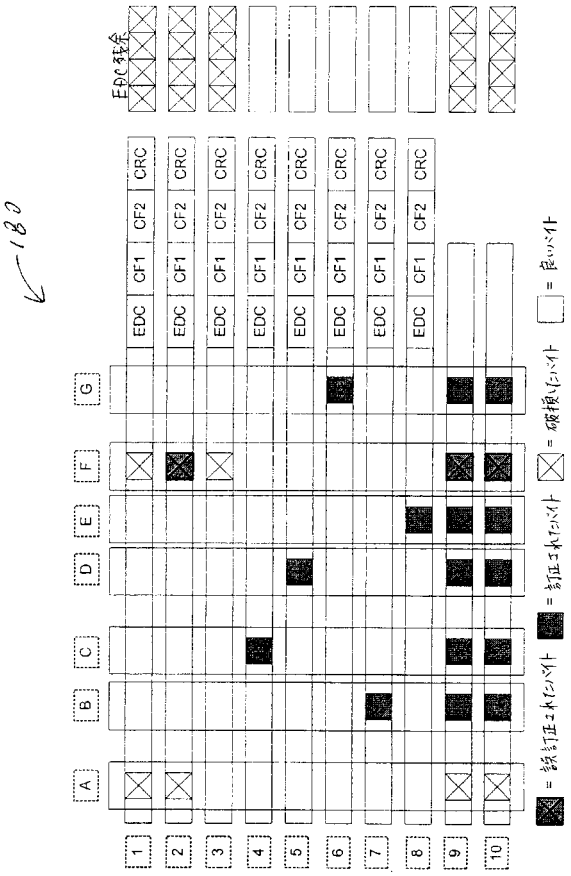
【図１６】



【図１７】



【図１８】

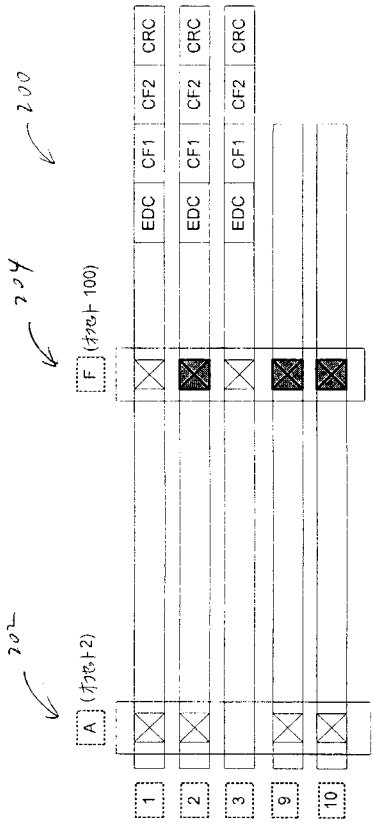


【図１９】

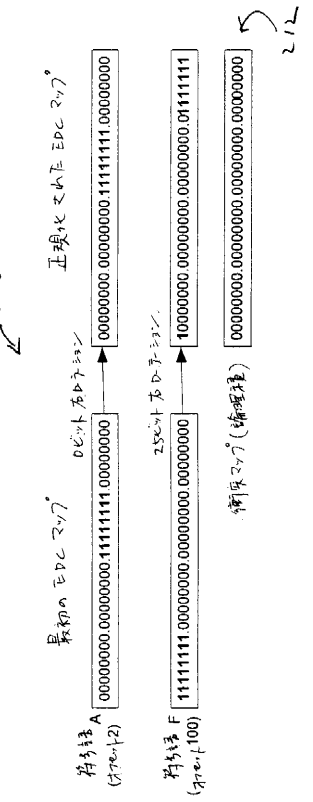
190

符号語	訂正されたブロック番号
B	7
C	4
D	5
E	8
F	2
G	6

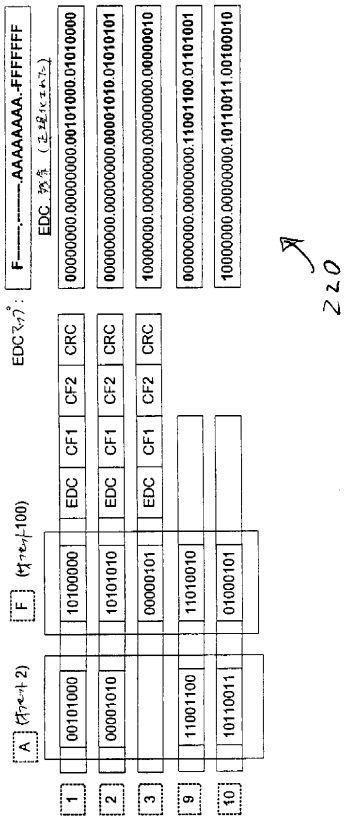
【図２０】



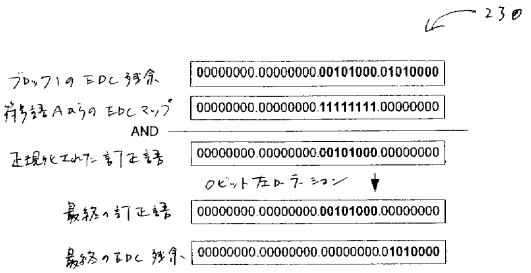
【図２１】



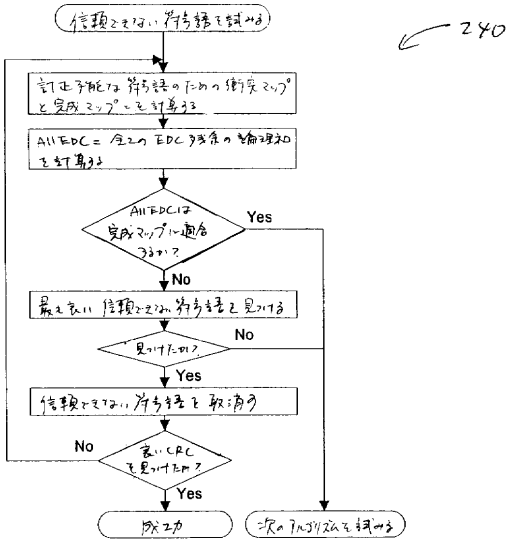
【図２２】



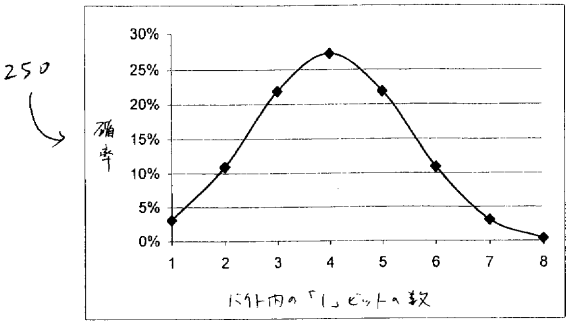
【図２３】



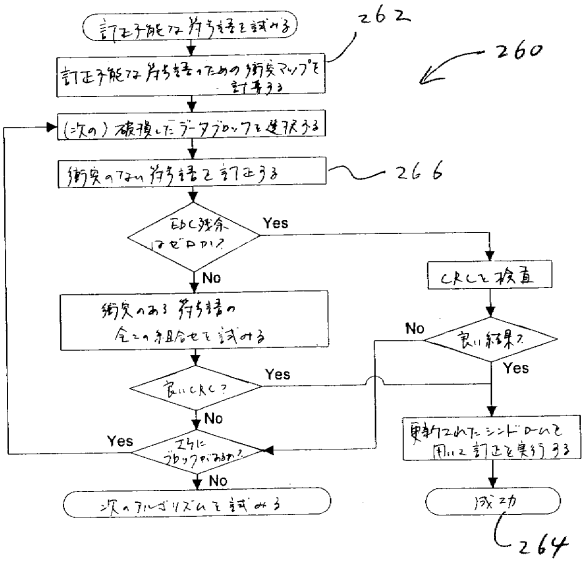
【図２４】



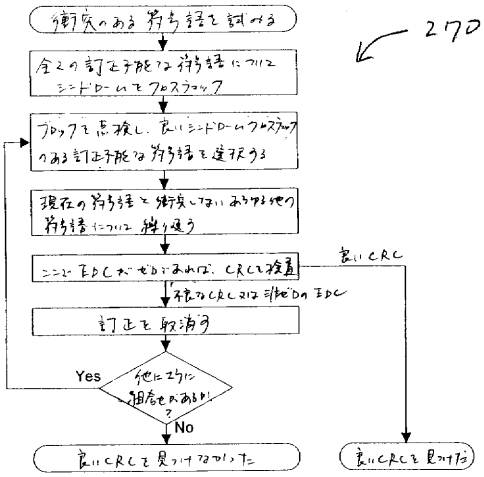
【図２５】



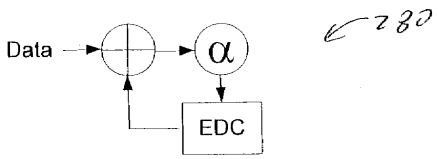
【図２６】



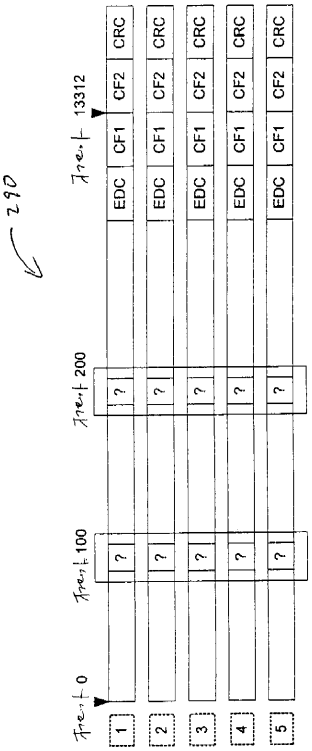
【図２７】



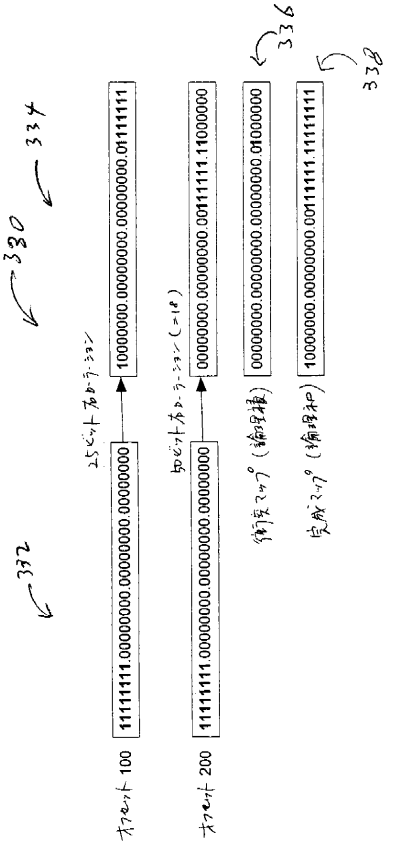
【図２８】



【図２９】



【図３０】



フロントページの続き

(51)Int.Cl.	FI		テーマコード（参考）
	Ｇ１１Ｂ 20/18	５７２Ｆ	
	Ｇ１１Ｂ 20/18	５７２Ｂ	
	Ｇ１１Ｂ 20/18	５７２Ｃ	
	Ｇ１１Ｂ 20/18	５３６Ｃ	
	Ｇ１１Ｂ 20/18	５３６Ｄ	

(74)代理人 100098316
弁理士 野田 久登
(74)代理人 100109162
弁理士 酒井 將行
(72)発明者 マシュー・ブイ・ボール
アメリカ合衆国、８０３０１ コロラド州、ボルダー、チザム・トレイル、３３１５、ナンバー・２０４

Ｆターム(参考) 5J065 AC03 AD04 AD11 AH20