

(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2009-22020

(P2009-22020A)

(43) 公開日 平成21年1月29日(2009.1.29)

(51) Int.Cl. F 1 テーマコード (参考)
H 0 4 N 5/225 (2006.01) H 0 4 N 5/225 F 5 C 1 2 2

審査請求 有 請求項の数 4 O L 外国語出願 (全 502 頁)

(21) 出願番号	特願2008-209323 (P2008-209323)	(71) 出願人	303024600
(22) 出願日	平成20年7月16日 (2008.7.16)		シルバーブルック リサーチ ビーティワ
(62) 分割の表示	特願2000-503510 (P2000-503510) の分割		イ リミテッド
原出願日	平成10年7月15日 (1998.7.15)		オーストラリア ニューサウスウェールズ
(31) 優先権主張番号	P08003		2 0 4 1, バルメイン, ダーリング ス
(32) 優先日	平成9年7月15日 (1997.7.15)	(74) 代理人	100083138
(33) 優先権主張国	オーストラリア (AU)		弁理士 相田 伸二
(31) 優先権主張番号	P08005	(72) 発明者	シルバーブルック キア
(32) 優先日	平成9年7月15日 (1997.7.15)		オーストラリア エヌエスタブリュ 2 0
(33) 優先権主張国	オーストラリア (AU)		4 0 レクハルト キャサリン ストリー
(31) 優先権主張番号	P08031		ト 2 1 4
(32) 優先日	平成9年7月15日 (1997.7.15)		
(33) 優先権主張国	オーストラリア (AU)		

最終頁に続く

(54) 【発明の名称】 向きの検知を伴うカメラデバイス、及び方法

(57) 【要約】 (修正有)

【課題】画像処理方法及び装置に関し、一体型カラープリンタ付きのデジタルカメラを提供する。

【解決手段】画像を検知するように適合されたエリアイメージセンサと、該検知された画像を保存するように適合されたメモリー記憶装置と、重力に関してカメラの向きを検知するように適合された向きセンサと、検知された画像に向き依存性の変換を適用することにより、画像を加工するように適合されたプロセッサと、画像をプリントメディアに印刷するための内蔵式プリンターと、を備え、向き依存性の変換は、(i) 検知されたカメラの向きを使用して、検知された画像中に少なくとも1つの顔を検出し、該検出された少なくとも1つの顔に関連して検知された画像を変更すること、(i i) 検知されたカメラの向きと調和する向きで、変換された画像を形成するために検知された画像中に文字情報を出すことのグループから選択される、カメラデバイス。

【選択図】 なし

【特許請求の範囲】

【請求項 1】

画像を検知するように適合されたエリアイメージセンサと、
該検知された画像を保存するように適合されたメモリー記憶装置と、
重力に関してカメラの向きを検知するように適合された向きセンサと、
変換された画像を形成するために前記検知された画像に向き依存性の変換を適用することにより、前記検知されたカメラの向きを利用して前記検知された画像を加工するように適合されたプロセッサと、
前記変換された画像をプリントメディアに印刷するための内蔵式プリンターと、
を備え、

10

前記向き依存性の変換は、

(i) 前記検知されたカメラの向きを使用して、検知された画像中に少なくとも 1 つの顔を検出し、前記変換された画像を形成するために、該検出された少なくとも 1 つの顔に関連して前記検知された画像を変更すること、

(i i) 前記検知されたカメラの向きと調和する向きで、前記変換された画像を形成するために前記検知された画像中に文字情報を出すこと

のグループから選択される、

カメラデバイス。

【請求項 2】

前記向き依存性の変換は、前記変換された画像を形成するために、前記検知された画像の中に特徴を変形させることを含む、

20

請求項 1 に記載のカメラデバイス。

【請求項 3】

画像を検知するように適合されたエリアイメージセンサと、
該検知された画像を保存するように適合されたメモリー記憶装置と、
重力に関してカメラの向きを検知するように適合された向きセンサと、
変換された画像を形成するために前記検知された画像を加工するように適合されたプロセッサと、

前記変換された画像を印刷するように適合された内蔵式プリンターと、
を備え、

30

(a) 前記向きセンサーによって、重力に関してカメラの向きを検知するステップ

(b) 前記プロセッサによって、前記変換された画像を形成するため、前記検知された画像へ向き依存性の変換を適用することにより前記検知されたカメラの向きを利用して前記検知された画像を加工するステップ

(c) 該変換された画像を前記内蔵式のプリンターによって印刷するステップ
を含み、前記向き依存性の変換は、

(i) 前記検知されたカメラの向きを使用して、検知された画像中に少なくとも 1 つの顔を検出し、前記変換された画像を形成するために、該検出された少なくとも 1 つの顔に関連して前記検知された画像を変更すること、

(i i) 前記検知されたカメラの向きと調和する向きで、前記変換された画像を形成するために前記検知された画像中に文字情報を出すこと

40

のグループから選択される、

デジタルカメラデバイスによってキャプチャされた画像を加工する方法。

【請求項 4】

さらに、前記向き依存性の変換は、

(i i i) 前記変換された画像を形成するために前記検知された画像中に特徴を歪曲させること、

(i v) 前記変換された画像を形成するために、画像中に登場する動物や人間の顔の特徴に関連する情報を配置すること

のいずれか、又は両方を含む、請求項 3 に記載の方法。

50

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、画像処理方法及び装置に関し、特に、画像処理機能を有するデジタルインスタントカメラを開示する。

【0002】

本発明は、さらに、デジタルカメラ技術の分野に関し、特に、一体型カラープリンタ付きのデジタルカメラを開示する。

【背景技術】

【0003】

従来のカメラ技術は、長年に亘り、感光性フィルムに映し出される画像のネガに依存する光学処理システムの設備に依存してきている。この感光性フィルムは、“定着(fix)”して、原画像の再生を行うポジプリントの作成を許容するために後で化学的処理される。そのような画像処理技術は、標準的になってきていたけれども、画像のフルカラー処理においては不経済で困難な技術が必要とされるため、過度に複雑になり得る。近年、デジタルカメラが利用可能になってきている。これらのカメラは、標準的には、際立った画像を検知するための電荷結合素子(CCD)の利用に依存している。該カメラは標準的には、続けて行う画像処理(manipulation)や印刷のためにコンピュータ装置に画像を転送するためのコネクタに加え、検知したシーンのストレージのためのストレージ・メディアを有する。

【0004】

そのような装置は、全ての画像がカメラによって保存されており、しばらく後でないと印刷できないという点で不便である。また、該カメラは、多数の画像の保存のために十分なストレージ能力を有していなければならず、加えて、カメラのユーザは、画像のダウンロードやコンピュータ・プリンターによる印刷等のためにコンピュータシステムにアクセスしなければならない。

【0005】

さらに、インスタント画像の生成を可能にするボラロイド(登録商標)タイプのインスタントカメラは、しばらくの間、使用されてきた。しかしながら、このタイプのカメラは、限定されたサイズの画像のみの作成に限られていて、また、化学物質の使用には多くの問題があり、特に、これらのタイプのカメラで作られた写真は劣化する。

【0006】

このような装置や他の画像キャプチャー装置を使用する場合、写真撮影の際、音声ひや他の環境の情報を適切に取り込めることが望ましい。

【0007】

さらに、左眼に与えられる第1画像と右眼に与えられる第2画像とで三次元面の錯覚を起こすという立体視の生成が、よく知られている。しかしながら、従前のシステム複雑な準備を必要とし、現物に極めて近い画像(ハイファイ画像)は通常不可能であった。さらに、画像の一般的な選択は、特別に準備された画像のみに限定されている。

【0008】

現物に極めて近い立体画像(ハイファイ立体画像)を要求に応じて生成できること、特に、立体画像を自由に撮影することのできるポータブル・カメラ装置によって画像を作成することが一般的に必要なである。

【0009】

さらに、カメラ装置によってキャプチャーされた画像を片面に有すると共に、もう片面に郵便料金払い済み印と住所を有する、自動的にカスタマイズされた葉書を、そのようなカメラ写真画像作成システムが作成できれば、非常に便利だろう。

【0010】

近年、画像の長い“パノラマ”ビューを作成する写真再生技術については、ますます一

10

20

30

40

50

般的になってきている。これらの画像は写真用紙などの上に作成される。また、さらなる“パノラマ”のタイプのビューを作成するため、画像の構成は、幅に比べて長さが長くされる。残念ながら、画像が形成される写真用紙は、小径のロールの状態で保存されていた場合に問題が発生する。

【0011】

近年、フィルターを配置して、人気のある芸術的な絵のスタイルに似せた効果を画像に生じさせるようにすることがかなり一般的になってきている。これらのフィルターは、画像を撮影すると共に、芸術的なスタイルの一つについての芸術的演出を最初の画像に与えて第2の画像を得るように設計されている。現代において極めて人気のある芸術家の一人は、ピンセント・ファン・ゴッホである。彼の絵画のフラット・エリアにおけるブラッシュ

10

【0012】

残念ながら、ワーピングのシステムは、ハイエンドコンピュータにて使用されることが一般的であり、画像をコンピュータの中にスキャンすることやプリントアウトすることは一般には不便である。携帯用カメラ等を使って画像を取得するような場合は、取得画像を

20

【0013】

さらに、多様な画法を真似るという効果に大きな価値があるとしばしば考えられる。さらに進んで、1つの簡単な形態の中にこれらの効果を組み入れることができれば、それらの効果は、デジタル撮影能力を有する携帯用カメラ装置と合体させるのに適切であり、それによって、カメラ装置で撮影されるシーンについて、所望のフィルター処理された画像を生成することができるだろう。

【0014】

残念ながら、今日 製造されるカメラにおいて厄介なシステムの要求である、デジタル画像変換技術やフィルター交換技術は、明らかに、未だ利用できない技術を利用することができず、今までのところ、創造も着想もできていないフィルターを利用することもできない。

30

【0015】

カメラ技術の極めて一般的な方式の一つは、伝統的なネガフィルムとポジプリント写真である。このケースでは、ネガフィルムにシーンを撮影するためにカメラが使用され、該ネガフィルムは陰画を定着させるために処理される。続いて一連のプリントがネガフィルムから作成される。さらに、一連のプリントはネガからどんなときにでも直ぐに作成されることができる。残念ながら、デジタルカメラ（本願にて提案されるものも含む）では、後においてさらなる画像のコピーが要求されるなら、キャプチャーされ印刷される写真を

40

【0016】

さらに、下塗りの際のメディアの選択に起因して色が制限され、風景についてのほとんど全ての芸術的絵画は限定的な色域（gamut）を使用している。このような制限は、様々な芸術的効果を生じさせるために芸術家によってしばしば使われる。この処理の古典的な

50

見本は、次のような周知の芸術作品を含む：

- － カミーユ・ピサロ “L'île Lacroix-Rouen、霧の効果” 1888年、フィラデルフィア美術館
- － シャルル・アングラン “セーヌ河、朝” 1889年、プティ・パレ・コレクション、ジュネーブ
- － アンリ・ヴァン・ド・ヴェルド “Crepuscule” 1892年、クレラー＝ミュラー国立美術館、オッテルロー
- － ジョルジョ・スーラ “バ＝ビュタンの砂浜、オンフルール” 1886年－トゥルネ美術館

【0017】

任意の入力画像から、上記リストのそれと似たような効果や特性を持つ出力画像を作成することが望ましい。

【0018】

何らかの衣服が創造されるとき多くの創造的な判断がなされる。初めに、衣服の形状やスタイルであり、加えて、布の色や種類である。しばしは、ファッションデザイナーは、衣服を完成させる前に、多くの異なる代替案にトライし、最終的なファッションの製品を描こうとする。

【0019】

このようなプロセスは、衣服の迅速かつ柔軟な変更や、ファッション製品の最終外観の迅速な判断機会のためには一般的に不十分である。

【0020】

双眼鏡や望遠鏡は良く知られている。特に、双眼鏡を例にすると、該装置は、ユーザの視力の能力を高めるために景色の望遠の拡大を提供する。さらに、夜間用双眼鏡等のような装置も、また、ユーザの視力の能力を高めるために使われる。残念ながら、これらのシステムは、リアルタイムの、アナログの光学部品に依存し、見えた景色の永久記録は困難である。シーンの永久コピーの記録に適していると思われる1つの方法は、シーンを取得するCCD等のようなセンサー装置を取り付けることと、後に行う印刷のためにストレージデバイスに保存することである。残念ながら、そのような構造は、野外で、携行性に富む状態での双眼鏡の使用が望まれる場合には特に邪魔となる。

【0021】

圧縮した状態での情報のストレージについては、多くの形態のものが周知である。例えば、コンピュータ装置の分野では、固定された、或いは携行できる磁気ディスク装置の使用は一般的である。携行できるディスクについては、“フロッピーディスク”や“ジップディスク”や、ポータブル磁気ストレージメディアの他の方式が、市場において大量に受け入れられている。携帯可能なストレージの他の形式は、一連の細長いピット（螺旋状のトラックに沿って配置されたものであり、レーザビーム装置によって読み取られるもの）を利用するコンパクトディスク“CD”である。CDの利用は、ストレージについての極めて低コストの形態を提供する。しかしながら、必要とされる技術はかなり複雑であり、書き換え可能なCDタイプの装置の使用は極めて限定される。

【0022】

ストレージの他の方式としては磁気カードがあり、しばしばクレジットカードや同種のものに使用される。これらのカードは、標準的には、カードのユーザに関連する情報を記録するため、裏面に磁気の帯を有している。近年、集積回路タイプの装置をカードに組み込むというスマートカード技術の形で、磁気カードの利便性が伸びてきている。残念ながら、そのような装置のコストはしばしば高く、使用される技術の複雑さもまた著しい。

【0023】

従来銀塩カメラの処理システムは、多数のプリントの作成のため、周知の“ネガ”の使用を生じさせてきた。そのネガは、標準的にはプリントの生産源となり、その実施は、連続使用のためのケアや保護のために成長してきた。

【0024】

フォールトトレラント方法にて検知される符号化システムでは、効率的な復号化が可能となる符号化がどのようにすれば可能かという、重要な懸案がある。したがって、効率的な符号化システムの提供が望まれる。

【0025】

本発明の概要

本発明は、一体型カラープリンタ付きデジタルカメラを備えたカメラシステムの代替形態の提供に関する。加えて、該カメラは、イメージ・センシング・システムの見かけの解像度を増加させ、様々な“芸術的なスタイル”やグラフィック強化 (graphic enhancement) に画像を変換するためのハードウェア及びソフトウェアを備える。

【0026】

本発明のさらなる態様に従って、シーンを撮像するための少なくとも1つのエリアイメージセンサーと、前記撮像されたシーンを所定のシーン変換要求に従って処理するためのカメラ処理装置 (camera processor means) と、前記処理された画像のシーンをプリントメディアに印刷するためのプリンターと、前記カメラシステム内部の一つの取り外し可能なモジュールに収納されたプリントメディア及び印刷用インクと、を備えたカメラシステムが提供される。該カメラシステムは、前記エリアイメージセンサーによるシーンの撮像、及び前記プリンターによって前記シーンを印刷したものを前記カメラシステムから直接出力するためのハンドヘルドユニットを備える。

【0027】

前記カメラシステムは、好ましくは、プリントメディアの保管のためのプリントロールと、前記プリンターにより使用される印刷用インクを備えている。該プリントロールは前記カメラシステムから取り外し可能である。さらに、該プリントロールが前記カメラシステムに挿入されたときに認証が実行されるように、該プリントロールは、認証情報を含む認証チップを備え、前記カメラ処理装置は該認証チップを確認 (interrogate) するように適合されている。

【0028】

さらに、前記プリンターは、ドロップ・オン・デマンドのインクプリンターと、印刷された写真の分離のための裁断装置とを含む。

【0029】

本発明の第1態様に従って、シーンを撮像するための少なくとも1つのエリアイメージセンサーと、前記撮像されたシーンを所定のシーン変換要求に従って処理するためのカメラ処理装置と、前記処理された画像のシーンをプリントメディアに印刷するためのプリンターと、前記カメラシステム内部の一つの取り外し可能なモジュールに収納されたプリントメディア及び印刷用インクと、を有するカメラシステムが提供される。該カメラシステムは、前記エリアイメージセンサーによるシーンの撮像、及び前記プリンターによって前記シーンを印刷したものを前記カメラシステムから直接出力するためのハンドヘルドユニットを備える。

【0030】

好ましくは、該カメラシステムは、前記プリンターにより使用されるプリントメディア及び印刷用インクを収納するためのプリントロールを備えている。該プリントロールは前記カメラシステムから取り外しが可能である。さらに、該プリントロールが前記カメラシステムに挿入されたときに認証が実行されるように、該プリントロールは、認証情報を含む認証チップを備え、前記カメラ処理装置は該認証チップを確認 (interrogate) するように適合されている。

【0031】

さらに、前記プリンターは、ドロップ・オン・デマンドのインクプリンターと、印刷された写真の分離のための裁断装置とを含む。

【0032】

本発明のさらなる態様に従って、装置を操作するためのユーザ・インターフェースが提供される。該ユーザ・インターフェースは、装置に挿入されるカードを有しており、該カ

10

20

30

40

50

ードの表面には、前記装置の出力に関して該カードが有する効果について目に見える説明が含まれている。

【0033】

本発明のさらなる態様に従って、

シーンを撮像するための少なくとも1つのエリアイメージセンサーと；

前記撮像されたシーンを所定のシーン変換要求に従って処理するためのカメラ処理装置と；

前記処理された画像のシーンをプリントメディアに印刷するためのプリンターと；

前記プリンターのための前記プリントメディアと印刷用インクとを収納するための取り外し可能なモジュールと；

を備えたカメラシステムが提供される。該カメラシステムは、前記エリアイメージセンサーによるシーンの撮像、及び前記プリンターによって前記シーンを印刷したものを前記カメラシステムから直接出力するためのハンドヘルドユニットを備える。

【0034】

本発明のさらなる態様に従って、

画像を検知するセンサー装置 (sensor means) と；

検知された画像を所定の処理要求に従って処理するための処理装置と；

もしあるならば、音声信号を検知画像に関連付けて一致させるための音声録音装置と；

前記カメラシステムで供給されるプリントメディアの第1エリアに処理された検知画像を印刷し、前記プリントメディアの第2エリアに前記音声信号の符号化されたバージョンを印刷するための印刷装置と；

を備えたカメラシステムが提供される。好ましくは、該検知画像は前記プリントメディアの第1面に印刷され、前記音声信号の符号化されたバージョンは該プリントメディアの第2面に印刷される。

【0035】

本発明に従って、

画像を検知するエリアイメージセンサー装置と；

該検知した画像を保管するための画像ストレージ装置と；

画像検知時におけるカメラの向きを検出する定位装置と；

検知したカメラの向きを使用して前記検知画像を処理するための処理装置と；

を備えたカメラシステムが提供される。

【0036】

本発明のさらなる態様に従って、

焦点調整技術を使用して画像をキャプチャーし；

画像における構成の位置の指示器として焦点設定を利用し；

該焦点設定に特有の効果を生成するために該焦点設定を利用して画像を処理し；

その画像を印刷すること；

からなるデジタル画像処理方法が提供される。

【0037】

好ましくは、前記画像はズーム技術を使用してキャプチャーされ、そのズーム設定は、前記画像の部分を処理するように発見的な方法で使用される。

【0038】

本発明のさらなる態様に従って、アイポジション検知装置付きデジタルカメラにて撮影された画像を処理する方法が提供される。該方法は、空間的に変化する sense (アイポジション情報に依存するもの) で画像を処理するため、検知画像にアイポジション情報を使用するステップを含む。

【0039】

前記ステップは、興味のあるエリアの位置を前記検知画像の中で特定するためにアイポジション情報を使用することを含む。その処理は、画像への吹き出しの配置や、画像への特定のゆがみの適用を含むことができる。代わりに、その処理は、前記アイポジション情

10

20

30

40

50

報のエリアに、より細かい画像のためにブラッシュストロークフィルターを適用すること
も含む。理想的には、前記カメラは、処理の結果を大体直ぐに印刷することができる。

【0040】

本発明のさらなる態様に従って、自動露出設定を有するデジタルカメラで撮影された画
像の処理方法が提供され、該処理方法は、検知画像を処理するために前記情報を使用する
ステップを含む。

【0041】

上述の、情報を使用するステップは、画像中の色が前記自動露出設定の計算に変換され
てなる補正画像を生成するため、画像中の色のリマッピングを決定するために自動露出設
定を使用することを含む。その処理は、露出設定が暗い状態を示すときは、リマッピング
される画像の色が、より深く、より鮮やかになるようにし、露出設定が明るい状態を示す
ときは、リマッピングされる画像の色が、より明るく、より飽和するようにする。

【0042】

最近、デジタルカメラはますます一般的になってきている。これらのカメラは、標準的
には、電荷結合素子（CCD）配列を使用した所望の画像の取得、並びに、撮像されたシ
ーンの電子ストレージメディアへの保存（続けて行う画像処理や印刷のために、コンピュ
ータシステムに後で行われるダウンロードのためのもの）を行う。標準的に、画像を印刷
するためにコンピュータシステムを使用するとき、洗練されたソフトウェアが、要求に従
って画像を処理するために使用される。

【0043】

残念ながら、そのようなシステムは、キャプチャーされた画像の重要な後処理を必要と
し、通常は、撮影された向きに画像を提示し、キャプチャーされた画像の必要な修正又は
要求された修正を実行するため、後処理工程に依存する。さらに、写真が撮影される際の
多くの環境情報が失われる。

【0044】

本発明のさらなる態様に従って、デジタルカメラ及びフラッシュを使用してキャプチャ
ーされた画像の処理方法が提供され、該方法は、以下のステップを含む。

(a) 前記フラッシュの使用に起因した、前記キャプチャーされた画像のゆがみ（ディス
トーション、distortion）の発見（特定）

(b) 前記ゆがみの影響を局部的に減少させるように前記画像の修正

【0045】

本発明の第2の態様に従って、キャプチャーされた画像におけるフラッシュのゆがみ（
distortion）の影響を減少させるデジタルカメラであって、

(a) 画像のキャプチャーのために画像をキャプチャーするデジタル画像キャプチャー装
置と；

(b) フラッシュが誘発した色のゆがみの、前記キャプチャー画像内での位置を探すため
のゆがみ位置決定装置と；

(c) 該ゆがみ位置決定装置及び前記デジタル画像キャプチャー装置に接続されて、前記
ゆがみの影響を減少させるように前記キャプチャー画像を処理するように適合された画像
ゆがみ是正装置と；

(d) 表示のために前記画像ゆがみ是正装置に接続されている表示装置と；
を備えたデジタルカメラが提供される。

【0046】

本発明のさらなる態様に従って立体写真画像を作り出す方法が提供され、該方法は、

(a) シーンを立体画像的に撮像するためにカメラ装置を使用し；

(b) レンズシステムが第2面に形成された透明なプリントメディアにおける第1面部分
の所定位置に、一体的に形成された画像のような立体撮影画像を印刷して、該印刷された
立体撮影画像の観察者の左右の眼が前記シーンを立体画像的に見るようにする；
ことを含む。

【0047】

本発明のさらなる態様に従って、プリントメディア及びインク供給装置が提供される。該供給装置は、インクと該インク付着用のプリントメディアとをプリント装置に供給するように適合されている。該供給装置は、メディア・フォーマーの上に巻かれた、ロール状のメディアを内部に有している。また、該供給装置は、前記メディア及びインク供給装置の内部に一体的に形成されると共に、インク及びプリントメディアの前記プリント装置への供給のために前記プリント装置に接続されるように適合された少なくとも1つのインク・リザーバを有している。

【0048】

本発明のさらなる態様に従って、カメラ画像システムにて使用されるプリントロールであって、フォーマットされた多数の葉書情報が、所定の間隔で裏面に印刷されてなるプリントロールが提供される。

10

【0049】

本発明の第2の態様に従って、カスタマイズされた葉書の作成方法が提供される。該作成方法は、次のステップを含む。すなわち、フォーマットされた多数の葉書情報区域を所定位置に有する裏面を備えるプリントロールを備えるカメラ装置の使用と、該プリントロールの対応する画像面へのカスタマイズされた画像の撮像と、該カメラ装置によって葉書を作成するための前記プリントロールの使用。

【0050】

本発明の第3の態様に従って、カメラ画像を有する葉書を郵便制度を通じて送る方法が提供され、該方法は、次のステップを含む。すなわち、前払いの郵便料金が含まれているプリントロールを売るステップと、葉書を作成するために該プリントロールをカメラシステムで使用するステップと、該前払いした葉書を郵便で送るステップ。

20

【0051】

本発明は、さらに、一体型プリンター装置付きカメラシステムを対象とする。そのプリントメディアは、カメラシステムから取り外し自在であり、該プリントメディアに関する重要な情報を保存するための装置を含む。

【0052】

本発明は、さらに、一体型プリンター装置付きカメラシステムを対象とする。そのプリントメディアは、カメラシステムから取り外し自在であり、該プリントメディアへの認証アクセスを提供する。

30

【0053】

本発明のさらなる態様に従って、使用に際してカールする度合いが低い、平らなプリントメディアが提供される。該プリントメディアは、前記平面の方向に異方性の剛性を有している。本発明の第2の態様に従って、異方性剛性を有する平らなプリントメディアにプリントされた画像においてカールを減少させる方法が提供される。該方法は、前記プリントメディアの部分に局所的な圧力を与えるものである。

【0054】

本発明はさらに、一体型プリンター装置を有するカメラシステムを対象とする。該カメラシステムは、プリンターに残っている画像の数のインジケータを有する。該インジケータは、多くの異なるモードで、残っているプリントの数を表示することができる。

40

【0055】

本発明のさらなる態様に従って画像の自動処理方法が提供される。該処理方法は、高い空間的な変化を持つ特徴を、画像の中で位置特定し、高い空間的な変化を持つそれらの領域から放射する一連のブラッシュストロークを画像に描くことを含む。好ましくは、そのブラッシュストロークは、重要な特徴付けるものの近傍ではサイズが小さい。加えて、ブラッシュストロークの所定部分の位置はジッターリングの下になる。

【0056】

本発明のさらなる態様に従って、入力画像を歪ませるようなワーピング方法が提供される。該方法は、所定の寸法A×Bである任意の出力画像のためのワーブマップであって、該ワーブマップ

50

の各要素は、理論上の入力画像における対応領域を、前記要素の座標位置に対応した前記任意の出力画像の画素位置にマッピングするワーブマップを入力するステップと；
縮小されたワーブマップを形成するためにワーブマップをワーブ画像の次元に縮小するステップと；

該縮小されたワーブマップの各要素のため、該要素の値と隣接する要素の値とから前記入力画像における貢献領域を計算するステップと；

該貢献領域から前記要素に対応するワーブ画像の画素のため、出力画像の色を決定するステップと；

を含む。

【0057】

本発明のさらなる態様に従って、センサー装置によって読み取るため、メディアに保存されたデータの復元力を増加させる方法が提供される。該方法は、

(a) 回復可能な方法で保存されているデータを、高周波成分を有する変調信号にて変調するステップと；

(b) 前記データを変調された形式で前記メディアに保存するステップと；

(c) 該変調され保存されているデータを前記センサー装置によって検知するステップと；

(d) 変調され前記メディアに保存されているデータの位置の分離を感知するため、変調され保存されているデータの変調を中立化するステップと；

(e) 保存データの内の変調されていないものを、保存データの内の変調されているものから回収するステップと；

を含む。

【0058】

本発明の好適な実施例は、カメラシステムの中に設けられたシステムであって、後で復号化が行えるように、CCDタイプの装置を介してデータを読み込むためのカード読み取りシステムに関して説明される。さらに、該好適な実施例の説明は、誤差制御システムの分野に大きく依存する。したがって、この仕様が案内される人は、誤差制御システムのエキスパートになるべきである。特に、“デジタル・コミュニケーションやストレージのための誤差制御システム”ステファン・B・ウィッカー著、プレティス・ホール社出版、1995年、特にリード・ソロモン・コードの考察のような標準的なテキストや他の標準的なテキストにしっかりと精通すべきである。

【0059】

本発明の目的は、スキャンされた画素を含むスキャンされた画像を、対応するビットマップイメージに変換する方法を提供することであって、該方法は、ビットマップイメージにおける各ビットのために、次のステップを有する。すなわち、

a. スキャンされた画像における周囲のビットの位置から現在のビットにおけるスキャンされた画像での予期された位置を決定するステップ、

b. 前記スキャンされた画像における期待された対応画素の位置での値から前記ビットの適当な値を決定するステップ、

c. 前記予期された位置での予期された強度の中心の重心の寸法を決定するステップ、

d. 前記スキャンされた画像における現在のビットの周囲に近接した画素のため、重心の寸法を決定するステップ、

e. 前記重心の寸法が前記予期された位置に関連して伸びる場合、伸びた重心の寸法を有する画素に該予期された位置を調整するステップ、
とを有する。

【0060】

本発明のさらなる態様に従って、

画像を検知できるデジタルカメラ装置と；

該画像を処理すべく該デジタルカメラ装置に挿入されて該デジタルカメラ装置に画像処理の指示（画像にテキストを付加する指示）を与えるように適合された画像処理データ入

10

20

30

40

50

力カードと；

前記画像に付加する新たなテキスト文字を作成するために、前記画像処理指示に関連して前記デジタルカメラ装置により使用される一連の非ローマ・フォント文字を有するテキスト入力装置であって、前記画像に付加する前記文字を入力するために前記デジタルカメラ装置に接続されるテキスト入力装置と；

を備えた、画像を編集する、テキストのための装置が提供される。

【0061】

望ましくは、前記新しいテキスト文字を作成するために前記画像処理指示に従って前記装置により要求された場合に、フォント文字が前記デジタルカメラ装置に送られる。前記画像処理データ入力カードは、ローマフォント文字のセットと、ヘブライ語、キリル文字、アラビア語、漢字及び中国文字 (chinese character) の少なくとも1つを含む非ローマ文字とを含む。

【0062】

本発明のさらなる態様に従って、

画像を検知できるデジタルカメラ装置と；

該画像を処理すべく該デジタルカメラ装置に挿入されて該デジタルカメラ装置に画像処理の指示（画像にテキストを付加する指示）を与えるように適合された画像処理データ入力カードと；

前記画像に付加する新たなテキスト文字を作成するために、前記画像処理指示に関連して前記デジタルカメラ装置により使用される一連の非ローマ・フォント文字を有するテキスト入力装置であって、前記画像に付加する前記文字を入力するために前記デジタルカメラ装置に接続されるテキスト入力装置と；

を備えた、画像を編集する、テキストのための装置が提供される。

【0063】

本発明の目的は、作り出される新しいフィルターの利点、及び、プリントアウトするデジタル画像の処理のための柔軟性のある方法の利点に加えて、アップデートされる技術の利点を利用することである。

【0064】

本発明のさらなる態様に従って、一連のインクドットからなる入力画像の再生のための画像コピー装置が提供され、該装置は、

(a) 対応した見本画像を作成するため、前記ドットの周波数よりも高いサンプリングレートで前記入力画像を撮像するイメージング・アレイ装置と；

(b) 前記見本画像中の印刷ドットの位置を決定するために前記画像を処理する処理装置と；

(c) 前記印刷ドットの位置に対応する、プリントメディア上の位置にインクドットを印刷するプリント装置と；

を含む。

【0065】

2. 前記コピー装置が前記入力画像のフルカラーコピーを印刷する、請求項1に記載された画像コピー装置。

【0066】

本発明のさらなる態様に従って、ボケ修正画像を出力するためのカメラシステムが提供される。該カメラシステムは、

画像を検知するイメージ・センサーと；

外部環境に関連した前記画像の何らかの動きを検知して、その動きを示す速度出力を生成するための速度検出手段と；

それらのイメージ・センサーや速度検出手段に接続されると共に、前記画像のボケ修正を行ってボケ修正画像を出力するように適合されたプロセッサ手段と；

を備えている。

【0067】

望ましくは、前記カメラシステムは、前記ボケ修正画像を直ぐに出力するために印刷装置に接続されるものであって、携帯用のハンドヘルドユニットである。前記速度検出手段は、微小電気機械（MEMS）装置のような加速度計を含むことができる。

【0068】

本発明のさらなる態様に従って、フォトセンサー読み取り形成品が提供され、該形成品は、

- (a) 発光素子の挿入のための、一連の発光体収納部と；
 - (b) 前記一連の発光素子から発光された光を、撮像されるべきオブジェクトの表面に焦点を合わせるための発光素子合焦装置と；
 - (c) 一連のフォトセンサー列を挿入するためのフォトセンサー収納部と；
 - (d) 撮像されるべきオブジェクトからの反射光を、前記CCD列の特徴的な部分に焦点を合わせる合焦装置と；
- を備える。

【0069】

本発明の態様に従って、コンピュータシステムに接続されるプリンター装置が提供される。該プリンター装置は、プリントメディアに画像を印刷するためのインクジェット・プリントヘッドからなるプリントヘッドユニットを備え、さらに、消耗財であるプリントロールユニットを挿入するための空洞を有する。該プリントロールユニットは、該空洞に挿入される消耗財であるプリントメディアとインクとを含み、該インクは前記プリントメディアに画像を印刷するため、前記インクジェット・プリントヘッドにより使用される。

【0070】

本発明のさらなる態様に従って、

画像を検知するセンサー装置（sensing means）と；

カメラに入力される修正指示に従って前記検知した画像を修正する修正装置と；

該修正された画像を出力するための出力装置と；

を備えたデジタルカメラシステムであって、前記修正装置が、中央のクロスバー・スイッチの周りに配列された一連のPE（Processing element）であるものが提供される。望ましくは、該PEは、書き込み可能なコントロール・ストアを有するマイクロコード・ストアの制御下で動作する算術論理演算ユニット（ALU）を有する。前記PEは、該素子にて使用される画素データを保存するための内部入出力FIFOを含むことができ、前記修正装置は、該装置へ画像の画素データを書き込んだり読み込んだりするのための読み書きFIFOに接続されている。

【0071】

それぞれのPEは輪になるように配置されることができ、また、それぞれのPEは最も近い隣のPEに単独で接続される。前記ALUは、内部クロスバースイッチを介してALU内の一連の中央演算装置に接続された一連の入力を受け、テンポラリーデータのストレージのための多くの内部レジスターを有する。該中央演算装置は、乗算器、加算器及びバレル・シフターの内少なくとも1つを有している。

【0072】

前記PEはさらに、該PEに画素データを転送するためのコモン・データバスに接続されている。該データバスはデータキャッシュに接続され、該データキャッシュは前記PEと、画像を保管するメモリースタアとの間で中間キャッシュとして機能する。

【0073】

本発明のさらなる態様に従って、カードに高ピッチレートで保存された検知画像データを、リアルタイムで短時間に復号化する方法が提供され、該方法は、前記画像データの初期位置を検出するステップと；

該画像データの対応するビットパターンを決定すべく、該画像データを復号化するステップと；

を含む。

【0074】

本発明のさらなる態様に従って、カードに高ピッチレートで保存されて回転やワーピングやマーキングがされる検知画像データをフォールトトレラント方法によって短時間で復号化する方法が提供され、該方法は、

前記画像データの開始の初期位置を決定するステップと；

前記ピッチレートよりも高いサンプリングレートで前記画像データを検知するステップと；

カラム処理によってカラム中の検知画像データを処理し、次のカラムの各ドットの中心（重心）の期待される位置をキープし、期待される次の重心の位置をアップデートするように、各カラムを処理する場合に該重心の微細な調整を行うステップと；

を有する。

10

【0075】

本発明のさらなる態様に従って、ドット配列からなり、画素配列を形成するため、該ドット配列のピッチ周波数よりも高いレートでサンプリングされた検知画像データのドットの値を正確に検出する方法が提供され、該方法は、

前記画素配列の中で、期待された重心位置に対応した、期待された中央画素を決定するステップと；

前記画素の位置を中心とするドットの値を出力するルックアップテーブルのインデックスとして、前記中央画素の検知値、及び、多くの隣接画素の検知値を使用するステップと；

からなる。

20

【0076】

本発明のさらなる態様に従って、カードに高ピッチレートで保存され、回転やワーピングやマーキングの効果を前提とする検知画像データのドットの位置を正確に決定する方法が提供され、該方法は、

カラム中の前記画像データをカラム形式によって処理するステップと；

画素の前記処理されたカラムのドットパターンを記録するステップと；

前記処理された画素の記録されたドットパターンから、期待されたドットパターンを最新のカラム位置にするステップと；

前記期待されたドットパターンを、前記最新のカラム位置での検知画像データの実際のドットパターンと比較するステップと；

30

その比較が所定の誤差内である場合に、新しい実際のドット位置を生成するため、前記期待されたドットパターンに適合させるため、実際のドット位置を前記最新のカラム位置に帰るように前記最新のカラム位置を使用するステップと；

次のカラムのドット位置を決定する際に、最新のカラム位置にて前記実際のドット位置を利用するステップと；

を含む。

【0077】

本発明のさらなる態様に従って、画像に絵画の効果を擬態するために画像バンプマップを結合させる方法が提供され、該方法は、

絵の具が塗られる表面を生じさせる、画像キャンバスバンプマップを規定し、

40

該表面に描かれるオブジェクトのペイントバンプマップを規定し、

前記画像キャンバスバンプマップによる、前記ペイントバンプマップの修正の程度を決定する剛性因子を使用し、最終的な結合バンプマップを作成するために前記画像キャンバスバンプマップと前記ペイントバンプマップとを結合すること、

を含む。

【0078】

本発明のさらなる態様に従って、芸術的な効果を生じさせるため、入力されてきた画像を自動的に処理する方法が提供され、該方法は、

目的とする芸術的效果を作成するため、所定の出力ガモットへの入力ガモットのマッピングを決定し、

50

所定の出力ガモットを有する出力画像に前記入力画像をマップするマッピングを利用する、
ことを含む。

【0079】

望ましくは、前記方法は、さらに、ブラッシュストローク・フィルタを使用した、出力画像の後処理のステップを含む。

【0080】

さらに、望ましくは、前記出力ガモットは、所定数の入力ガモット値を、対応する出力色ガモット値にマッピングすると共に、入力ガモット値から出力色ガモット値へのマッピングの維持を挿入することによって形成される。その挿入プロセスは、出力色ガモット値への入力ガモット値の所定数のマッピングの加重和を含む。

【0081】

本発明の第2の態様に従って、出力色ガモットに入るように入力色ガモットを圧縮する方法が提供され、該方法は、
現在の入力色の強さにてゼロクロミナンス値を決定するステップと；
前記ゼロクロミナンス値から前記入力色ガモットのエッジまでの距離であるソース距離を決定するステップと；
前記出力色ガモットの好適なエッジを決定するステップと；
前記ゼロクロミナンス値から前記出力色ガモットのエッジまでの距離であるターゲット距離を決定するステップと；
前記ソース距離と前記ターゲット距離との比から導き出される因子により、現在の入力色強さを計るステップと、
を備える。

【0082】

好ましくは、前記現在の入力色強さは、前記ゼロクロミナンス値ポイントからの前記現在の入力色の距離に基づく因子により計られる。

【0083】

本発明のさらなる態様に従って、カメラによって検知された画像の出力のための携帯用カメラが提供され、該カメラは、
画像を検知するためのセンサー装置（sensing means）と；
タイル状画像を作成すべく、検知された画像にタイル効果を付与するタイル装置と；
該タイル状画像を表示するための表示装置と；
を備えている。

【0084】

本発明のさらなる態様に従って、カメラによって検知された画像の出力のための携帯用カメラが提供され、該カメラは、
画像を検知するためのセンサー装置（sensing means）と；
テクスチャが付加された画像を作成すべく、検知された画像にテクスチャ効果を付与するテクスチャマッピング装置と；
該テクスチャが付加された画像を表示するための表示装置と；
を備える。

【0085】

本発明のさらなる態様に従って、カメラにより検知された画像を出力するための携帯用カメラが提供され、該カメラは、
画像を検知するためのセンサー装置（sensor means）と；
該検知された画像にて光を放つ光源の効果を真似たイルミネーション画像を作成すべく、検知画像に照明効果を付与する照明装置と；
該イルミネーション画像を表示するためのディスプレイ装置と；
を備えている。

【0086】

本発明のさらなる態様に従って、

検知画像の特徴を有する布製衣料を描くディスプレイ装置への出力のため、前記検知画像の処理のためにカメラそうちに入力される一連の入力トークン信号と；

前記出力画像を形成するために、前記入力トークン信号を読み、画像を検知し、読み込んだ入力トークン信号に従って該画像を処理するように適合されたカメラ装置と；

前記出力画像を表示するように適合されたディスプレイ装置と；

からなる衣類創作システムが提供される。

【0087】

本発明のさらなる態様に従って、画像センサー装置と画像表示装置とを有し、画像センサー装置によって検知された任意の画像の部分に衣類にマッピングして、前記画像表示装置にて表示する衣類創作システムが提供される。

10

【0088】

本発明のさらなる態様に従って、処理された画像を形成する方法が提供され、該方法は、一連のカメラ処理ユニット（camera manipulation unit）を接続することを含み、それぞれのカメラ処理ユニットは、処理された出力画像を形成すべく、入力画像に画像処理を与えるものである。それらのカメラ処理ユニットの最初の1つは周囲から画像を検知し、それらのカメラ処理ユニットの少なくとも最後の1つは、色落ちしない出力画像を形成する。

【0089】

本発明のさらなる態様に従って、遠い所にある被写体を眺めるための携帯用撮像装置が提供され、該撮像装置は、

20

遠く離れたところに見える被写体を拡大するための光学レンズシステムと；

該被写体を同時に検知するための検知システムと；

該検知された画像を処理してプリンター装置に送るために前記検知システムに接続される処理装置と；

要求に応じて前記携帯用撮像装置により検知された画像をプリントメディアに印刷するため前記処理装置に接続されたプリンター装置と；

を備えている。

【0090】

好ましくは、前記システムは、さらに、前記プリンター装置にロール状のプリントメディアとインクとを供給するため、前記プリンター装置に接続される、取り外し可能なモジュールに配置された、取り外し可能なプリントメディア供給装置を有する。

30

【0091】

前記プリンター装置は、検知した画像を出力するための、フルカラープリンターを提供する、インクジェットプリント装置を備えることができる。

【0092】

さらに、好ましい実施例は、遠い所の被写体を前記検知システムに投影するためのビームスプリッタ装置を備えた双眼鏡システムとして実施される。

【0093】

本発明のさらなる態様に従って、カードに印刷された一連のドットとして、フォールトトレラント方法により符号化されている録音済み音声を再生するシステムが提供される。

40

該システムは、

録音済み音声の目に見える形式のスキャンのためのオプティカル・スキャナー装置と；

スキャンされた音声符号を復号化して音声信号を生成するように、該オプティカル・スキャナー装置に接続された処理装置と；

要求に応じて該音声信号を再生するように、該処理装置に接続された音声エミッター装置と；

を備える。

【0094】

50

前記符号化は、録音済み音声のリード・ソロモン符号化を含むことができ、スキニングを補助するために高周波変調されたインクドット列を含むことができる。

【0095】

前記システムは、前記カードを挿入するための細長い穴を有する棒状のアームを備える。

【0096】

本発明のさらなる態様に従って、印刷されたカードに情報を担持させる方法を提供するものであって、該方法は、

カードの表面を多くの所定のエリアに分割するステップと；

該所定のエリアの内の第1エリアに格納されるべき第1のデータを印刷するステップと；

前記カードに保存されている情報を読み取るときに前記印刷された第1の所定エリアを使用するステップと；

該カードに保存された情報をアップデートするときに、該カードに保存されるさらなる情報が印刷される、データの印刷に使用されていない第2の所定エリアを決定するステップと；

を有する。

【0097】

望ましくは、それらのエリアは、予め決められた順番で選択される。また、例えば、データのリード・ソロモン符号化によってもたらされるフォールトレラントを有するデータを印刷するため、高解像度のインクドットプリンターを使用して印刷される。前記エリアのボーダーの線引きをするボーダー領域が印刷されて提供され、さらに、該ボーダー領域の位置を示すのに役立つ、多数のボーダー・ターゲット・マーカーが提供される。それらのボーダー・ターゲットは、第1色の大きなエリアと、該エリアの中心に配置される、第2色の小さなエリアとからなる。

【0098】

望ましくは、前記データは、市松模様パターンのような高周波変調信号を使用して印刷されている。その印刷されたものは、1インチにつき約1200ドットよりも大きい解像度のドット配列であり、好ましくは、1インチにつき少なくとも1600ドットよりも大きい解像度を有するドット配列である。該所定のエリアは、該カードの表面にて標準の配列に配列することができ、該カードは、一般的な長方形のクレジットカードのようなサイズ及び形状にすることができる。

【0099】

本発明のさらなる態様に従って、画像処理のための一連の指示を作成する方法が提供される。該方法は、

ユーザが選択するため、見本画像の初期配列を表示するステップと；

該見本画像の少なくとも1つをユーザが選択することを受諾するステップと；

見本画像のさらなる配列を作成するため、その選択結果を利用するステップと；

ユーザが最終的な最適画像を少なくとも1つ選択するまで、上述のステップを繰り返して適用するステップと；

前記一連の指示に従って見本画像を作成するステップと；

該一連の指示を出力するステップと；

を含む。

【0100】

さらに、前記方法は、ユーザの写真をスキャンすることと、各見本画像の作成のための初期画像として該スキャンされた写真を使用すること、を含む。前記指示は、前記カードの第2面に視覚表示されるように印刷されるのに加えて、第1面にエンコード形式で印刷されることができる。加えて、処理画像自体は印刷される。

【0101】

前記配列を形成するため、遺伝的アルゴリズムや遺伝的プログラミング技術を含む画像の形成に様々な技術が使用され得る。さらに、さらなる画像の形成において、‘これまで

10

20

30

40

50

のところベスト (best so far) ' の画像が使用のために保存される。

【0102】

望ましくは、カードや写真を販売する自動販売機に組み込まれたコンピュータシステムにこの方法が実行される。

【0103】

本発明のさらなる態様に従って、挿入されたカードに情報を保存するための情報保存装置が提供され、該装置は、
カードの面に印刷されたパターンであって、カードの所定数の可能動作領域 (possible active area) に配置されたパターンを検知するセンサー装置と；

該検知したパターンを対応するデータに復号化するためのデコード装置と；

カード表面であって前記活動領域の1つにドットパターンを印刷するための印刷装置と；
前記センサー装置及び前記印刷装置の公知の相対位置に、検知されるカードを配置する位置決め装置と；

を備え、

前記検知装置は、前記カードの最新の動作プリント領域にて印刷されたパターンを検知するように適合され、

前記デコード装置は、該検知したパターンを対応する最新のデータに復号化するように適合化され、

前記印刷装置は、最新のデータのアップデートが要求された場合に、カードの正確な位置決めのために位置決め装置が駆動された後、前記活動領域の内の最新の1つに、アップデートされた最新データをプリントするように適合される。

【0104】

前記印刷装置は、望ましくは、1度にカード幅の印刷が可能なカード幅プリントヘッドからなるインクジェットプリンタ装置を備える。前記位置決め装置は、カードを挟んでカードの移動をコントロールするための一連のピンチローラーを備えることができる。前記印刷されたパターンは、例えば、リードソロモン復号化を使用して、フォールトトレラント方法により配列され、前記デコード装置は、該フォールトトレラントパターンのために好適なデコーダを有する。

【0105】

本発明のさらなる態様に従って、

画像を検知するイメージ・センサーと；

検知画像及び関連するシステム構成を保存するためのストレージ装置と；

前記検知画像を修正するために画像修正データモジュールを入力するためのデータ入力装置と；

前記検知画像の処理に加えて前記カメラシステムのコントロールのために、前記イメージ・センサーや前記ストレージ装置や前記データ入力装置に接続された処理装置と；

供給されるプリントメディアに、要求に応じて前記検知画像を印刷するための印刷装置と；

を備えたデジタルカメラシステム、並びに、

さらなる画像修正モジュールの挿入に基づき、前記デジタルカメラシステムの操作の修正を前記処理装置にさせるように適合された、画像修正データモジュールの提供方法が提供される。

【0106】

望ましくは、符号化されたデータをその表面に有するカードを前記画像修正データモジュールが有し、そのデータの符号化は印刷により行われ、前記データ入力装置は、カード表面をスキャンするための光学スキャナーを有する。操作の修正は、繰り返し、同じ画像に、一連の画像修正モジュールが挿入される順にそれぞれ画像修正を行う。

【0107】

本発明のさらなる態様に従って、

画像を検知するイメージ・センサーと；

検知画像及び関連するシステム構成を保存するためのストレージ装置と；

前記検知画像を修正するために画像修正データモジュールを入力するためのデータ入力装置と；

前記検知画像の処理に加えて前記カメラシステムのコントロールのために、前記イメージ・センサーや前記ストレージ装置や前記データ入力装置に接続された処理装置と；

供給されるプリントメディアに、要求に応じて前記検知画像を印刷するための印刷装置と；

を備え、画像修正データモジュールは、デジタルカメラシステムにて一連の診断テストの実施、及び前記印刷装置を介した結果の印刷を前記処理装置が行うように適合されたデジタルカメラシステムが提供される。

10

【0108】

望ましくは、符号化された指示データをその一面に有するカードを前記画像修正データモジュールが有し、前記処理装置は、該カードに符号化された指示データを読み取るための装置を有する。前記診断テストは、連続した全てが黒の帯を印刷するなどによって、プリンター装置の操作を向上させるため、プリンター装置のためのクリーニングサイクルを有することもできる。代わりに、インクジェットプリンターの操作を向上させるため、前記診断テストがノズルの調節を含むこともできる。加えて、カメラシステムの様々な内部操作パラメータを印刷することもできる。重力衝撃センサーをカメラシステムに装備した場合には、前記診断テストは、該センサーの局地を印刷することもできる。

【0109】

20

本発明のさらなる態様に従って、画像を生成するためのカメラシステムが提供され、該カメラシステムは、画像を検知するセンサーと；何らかの所定の処理要求に従って前記検知画像を処理するための処理装置と；プリントメディアの面に前記検知画像を印刷するための印刷装置と；前記プリントメディアに形成された磁気感受性面に関連情報を記録するための磁気記録装置とを備える。前記関連情報は、検知画像に関連した音声情報を含み、前記印刷装置は、望ましくは、検知画像を前記プリントメディアの第1面に印刷し、前記磁気記録装置は前記プリントメディアの第2面に前記関連情報を記録する。前記プリントメディアは、前記カメラシステムの内部の取り外し可能なロールに収納されることができる。1つの実施例では、前記磁気感受性面は、前記プリントメディアの裏面に貼り付けられた帯片を含むことができる。

30

【0110】

本発明のさらなる態様に従って、携帯用カメラ装置のイメージセンサーにキャプチャーされた画像の、色落ちしないコピーを作る方法が提供される。前記携帯用カメラ装置は、該カメラ装置に格納されているプリントメディアに印刷するための一体型コンピュータ装置と一体型プリンター装置とを備えており、前記方法は、前記イメージセンサーにて画像を検知するステップと；フォールトトレラントの符号性質を持つ符号化形式に前記画像を変換するステップと；前記一体型プリンター装置を使用して画像の色落ちしない記録としての画像の符号化形式を印刷するステップとからなる。

【0111】

40

好ましくは、前記一体型プリンター装置は、プリントメディアの第1面及び第2面に印刷するための装置を備え、前記検知画像やその視覚的な処理は第1面に印刷され、符号化された形式は第2面に印刷される。検知画像のサムネイルは、前記画像のエンコード形式の横に印刷されることができる。また、そのフォールト・トレラント符号化は、永久的な録音が、反復可能な高周波スペクトラル成分を有する符号化形式に市松模様のような高周波変調信号を適用するのに加えて、画像のリードソロモン符号化バージョンを形成することを含む。

【0112】

本発明の第1の態様に従って、カメラ装置で使用される画像処理カードの販売のための販売システムが提供される。該カメラ装置は、画像処理カードの挿入のためのカード処理インターフェースを有する。該画像処理カードは、前記カメラ装置での画像の処理のため

50

のものである。前記販売システムは、
画像処理カードを出力するための複数の印刷装置と；
該画像処理カードの作成のために必要な一連の画像処理カードデータのストレージのために、それぞれの印刷装置が対応するコンピュータシステムに接続されている。

【0113】

望ましくは、前記コンピュータシステムは、コンピュータ・ネットワークを介してキャッシュされる方法で、一連の画像処理カードデータを保存する。そして、カード配布コンピュータは、新しい画像処理カードをコンピュータシステムへ配布する。

【0114】

前記画像処理カードにイベントカード（季節ごとのイベントに利用すべくカードをプリントアウトするため、前記コンピュータシステムに分配される季節ごとのカード）が含まれる場合、本発明は特定用途を有する。

【0115】

本発明のさらなる態様に従って、オブジェクトの表面に符号化された、一連のブロックデータ領域を有するデータ構成が提供され、それぞれのブロックデータ領域は、符号化された形式に復号化されるデータを含む符号化データ領域と；

該符号化データ領域の第1の周辺部を囲むように配置された、一連のクロックマーク構成と；

該符号化データ領域の第2の周辺部を囲むように配置された、一連の、簡単に特定できるターゲット構成と；

からなる。

【0116】

前記ブロックデータ領域は、さらに、前記符号化データ領域の第3の周辺部の周りに配置された方向データ構成（orientation data structure）を含む。該方向データ構成は、前記周辺部のエッジに沿った、等データ点の線（a line of equal data points）を含むことができる。

【0117】

前記クロックマーク構成は、符号化データ領域のエッジに沿って配置されたデータ点の実質的近接第2の線（a substantially adjacent second line）に加えて、等データ点の第1の線を有することができる。該クロックマーク構成は、符号化データ領域の反対側に配置されることができる。

【0118】

前記ターゲット構成は、間隙を有するように配置された一連のデータポイントブロック（data points blocks）のセットを有し、該データポイントは、中央部以外では、第1の大きさ（magnitude）の定値を有し、中央部では、前記第1の大きさとは逆の大きさを有する。前記ブロックのセットは、さらに、前記逆の大きさの連続群の値を有する、ターゲット数表示構成を含む。

【0119】

前記データ構成は、理想的には基材表面に印刷された一連のドットにて活用される。

【0120】

本発明の第2の態様に従って、オブジェクトの表面に符号化されたデータ構成を復号化する方法が提供され、該データ構成は一連のブロックデータ領域を有し、それぞれのブロックデータ領域は、

符号化された形式に復号化されるデータを含む符号化データ領域と；

該符号化データ領域の第1の周辺部を囲むように配置された、一連のクロックマーク構成と；

該符号化データ領域の第2の周辺部を囲むように配置された、一連の、簡単に特定できるターゲット構成と；

からなり、前記方法は、

(a) 前記データ構成をスキャンするステップと；

10

20

30

40

50

- (b) 前記データ構成のスタートの位置を探すステップと；
 - (c) 前記ターゲット構成の最新の方向を判断して該ターゲット構成の位置を探すステップと；
 - (d) 前記ターゲット構成の位置から前記クロックマーク構成の位置を探すステップと；
 - (e) 前記符号化データ領域のビットデータの期待される位置を決定するために前記クロックマーク構成を使用するステップと；
 - (f) 各ビットデータのために期待されるデータ値を決定するステップと；
- を有する。

【0121】

前記クロックマーク構成は、符号化データ領域のエッジに沿って配置されたデータ点の実質的近接第2の線（a substantially adjacent second line）に加えて、等データ点の第1の線を有することができる。前記(e)の、使用するステップは、前記クロックマーク構成中の最新の位置を維持するため、疑似位相ロックループタイプのアルゴリズムを使用してデータポイントの第2の線に沿ってランすることも含む。

【0122】

さらに、決定するステップ(f)は、
検知したビット値を3つの隣接する領域（すなわち、中央領域、第1のlower extreme 領域、及び第2のupper extreme 領域）に分けることを含む、
第1の領域にそれらの値が含まれ、該第1の領域に対応するビット値を決定し；
第2の領域にそれらの値が含まれ、該第2の領域に対応するビット値を決定し；
前記中央領域にそれらの値が含まれ、その値が第1の値か第2の値かを決定するために空間的に囲まれる値を使用する。

【0123】

本発明のさらなる態様に従って、検知データの出力データ値を決定する方法が提供され、該方法は、

- (a) 検知されたデータ値を3つの隣接する領域（すなわち、中央領域、第1のlower extreme 領域、及び第2のupper extreme 領域）に分けることを含む、
第1の領域にそれらの値が含まれ、該第1の領域に対応するビット値を決定し；
第2の領域にそれらの値が含まれ、該第2の領域に対応するビット値を決定し；
前記中央領域にそれらの値が含まれ、その値が第1の値か第2の値かを決定するために空間的に囲まれる値を使用する。

【0124】

本発明のさらなる態様に従って、複数の異なる供給スロットに複数の異なる流体を供給するための流体供給装置が提供される。該供給スロットは、交互に配置するという方法で周期的な間隔を置いて配置され、前記流体供給装置は、
複数の異なる流体のそれぞれのための流体注入口装置と；

異なる流体のそれぞれのための流路であって、前記供給スロットのそれぞれに接続されたメインチャンネル流路と；

前記供給スロットのそれぞれを、対応するメインチャンネル流路に接続するサブチャンネル流路と；

を備えている。流体の数は2よりも多く、成形された流体供給ユニットの第1面に少なくとも2本のメインチャンネル流体装置が形成され、他のメインチャンネル流体装置は、成形品の表面を貫通する貫通孔を介して前記スロットに接続されたサブチャンネル流路で該成形品の上面に形成される。

【0125】

望ましくは、前記供給装置は、プラスチック射出成形されており、前記スロットのピッチレートは、1インチ当たり1000に等しいか、1000よりも少ない。さらに、スロ

10

20

30

40

50

ットの集合が、写真の幅に形成される。望ましくは、前記流体供給装置は、1又はそれ以上のピンチローラーを収容するためのローラースロット装置を複数有しており、前記流体はインクからなり、前記ローラーは、前記スロットに接続されたプリントヘッドをプリントメディアが横切るように制御される。前記スロットは、カラムに配置された一連のカラースロットに分割される。

【0126】

好ましくは、前記流体供給装置の少なくとも1つのチャンネルは、製造時に露出され、該露出面をシーリングテープにてシールされている。有利なことには、前記流体供給装置は、テープによる自動結合されたワイヤを得るため、TABスロットにより提供される。

【0127】

本発明のさらなる態様に従って、インク供給路を介して供給するインク吐出装置を少なくとも1つ使用する画像印刷用プリンター装置が提供され、該装置は、前記プリンターへのインク供給のためのインク供給装置に連動するように適合された、1つの出力色につき1つで一連のインク供給口と；

前記プリンター装置の外表面に沿った、一連の導電性コネクタパッドと；

インクを吐出させる一連のインク吐出装置を有するページ幅プリントヘッドと；

前記インク供給口から前記ページ幅プリントヘッドの前記インク吐出装置にインクを供給するインク供給システムと；

前記ページ幅プリントヘッドを前記導電性コネクタパッドに接続する、複数の相互接続ワイヤと；

を備え、

前記プリンター装置は、ハウジング装置の中に取り外し可能に挿入されるように適合され、該ハウジングは、前記導電性コネクタパッドとの相互接続のための相互接続部や、前記インク供給装置によるインクの供給のため、前記インク供給口との相互接続のためのインク供給コネクタを有している。

【0128】

望ましくは、前記複数の相互接続ワイヤは、前記プリンター装置の外表面を包み込むと共に、導電性コネクタパッドに接続されたテープ自動ボンディング・シートを形成する。

該相互接続されたワイヤは、

前記導電性コネクタパッドに相互接続されると共に、プリントヘッドの全長に沿って互いにほぼ平行な配置された第1のセットのワイヤーと、

該プリントヘッドの表面から互いにほぼ平行に配置された第2のセットのワイヤーと、

を有している。第1のセットのワイヤーのそれぞれは、第2のセットのワイヤの多くに接続されている。前記インク供給口は、薄いダイアフラム部を有している。該ダイアフラム部は、前記インク供給コネクタが前記ハウジング装置に挿入されることに基づき穴が開けられている。

【0129】

前記ページ幅プリントヘッドは、それぞれに所定数のインク吐出装置を有する、実質的に等しく繰り返し可能なユニット (substantially identical repeatable unit) を多数備えることができる。該ユニットは、所定数の接続ワイヤを有する標準インターフェースを有し、各標準インターフェース装置は、導電性コネクタパッドにグループで接続される。該プリントヘッド自体は、シリコンウェハから作成されることができ、ページ幅ストリップに分離される。

【0130】

本発明のさらなる態様に従って、電流の変動をモニターするという方法によって集積回路のモニタリングに対する抵抗を与える方法が提供され、前記方法は、前記集積回路の一部であるスプリアスノイズ発生回路を形成するステップを含む。

【0131】

前記ノイズ発生回路はLFSR (リニア・フィードバック・シフト・レジスタ) のような乱数発生器を含むことができる。

10

20

30

40

50

【0132】

本発明のさらなる態様に従って、低消費電力のCMOS回路が提供される。該回路は、第1のクロックと入力とに接続されたゲートを有するpタイプのトランジスタ、及び、第2のクロックと入力とに接続されたnタイプのトランジスタを備え、該CMOS回路は、オーバーラップしない方法により前記第1及び第2のクロックが切り換えられることに稼動される。

【0133】

該回路は、高出力スイッチング特性 (high power switching characteristics) を持つ第2の回路に実質的に隣接されることができる。該第2の回路は、ノイズ発生回路を有することができる。

10

【0134】

本発明のさらなる態様に従って、異なる出力状態に対応する複数のレベル状態を有する、メモリ回路のモニタリングに対する抵抗を与える方法が提供され、該方法は、有効な出力状態のためだけに中間状態を利用することを含む。該メモリはフラッシュメモリにすることができ、さらに、1かそれ以上のパリティビットを有することができる。

【0135】

本発明のさらなる態様に従って、集積回路に手を加える攻撃 (attempt) をモニターするため、ランダムノイズ発生器に取り付けられた回路 (circuit path) を使用する、前記集積回路に手を加えることに対する抵抗を与える方法が提供される。

【0136】

前記回路 (circuit path) は、互いに逆であって、様々な回路に接続され、リセット出力信号を生成するための排他的OR回路となる、第1パス及び第2パスを有する。

20

【0137】

本発明の第2の態様に従って、接地された大きな抵抗に一端が接続され、電源に接続された第2の大きな抵抗に他端が接続されたタンパー検出ラインが提供される。該タンパー検出ラインは、さらに、期待される電圧が所定の許容範囲内にあるかを比較するコンパレータに接続される。一連のテストに接続される抵抗の間には、そのテストの1つによってタンパリングが検出された場合にコンパレータがリセット信号を出力するように大きな抵抗が出力される。

30

【0138】

本発明のさらなる態様に従って、認証されるべき付属ユニットの有効性を判断するための認証システムが提供され、該システムは、第1及び第2のセキュアキーの確認のための中央システムユニットと；

該中央システムユニットに取り付けられた第1及び第2のセキュアキーを保持するオブジェクトと；

を備え、

第2のセキュアキーを保持するオブジェクトは取り外せないような方法で取り付けられ、前記中央システムユニットは、

第1のレスポンスを決定するために前記第1のセキュアキーを保持するオブジェクトに問い合わせを行い、

40

第2のレスポンスを決定するために前記第2のセキュアキーを保持するオブジェクトに問い合わせを行うために前記第1のレスポンスを利用し、

さらに、

前記第2のセキュアキーを保持するオブジェクトが、有効な取り付けユニットに取り付けられているかどうかを判断するために、前記第1のレスポンスと前記第2のレスポンスを比較するように適合されている。

【0139】

前記取り付けユニットの所定の使用の後に該ユニットが作動を止めるように、前記第2のセキュアキーを保持するオブジェクトは、さらに、効果的に単調減少するマグニチュー

50

ド・ファクターを有するレスポンスを含む。したがって、取り付けられたユニットは、消耗品を含むことができる。

【0140】

さらに、前記中央システムユニットは、前記第1のレスポンスを受け取った数を前記第1のセキュアキーを保持するオブジェクトに問い合わせることができ、そのとき、該中央システムユニットは、第2のレスポンスの判断のための前記第2のセキュアキーを保持するオブジェクトへの問い合わせに際して前記第1のレスポンスを使用し、該中央システムユニットは、前記第2のレスポンスの有効範囲を判断するため、前記第1のセキュアキーを保持するユニットに問い合わせを行うために前記第2のレスポンスを使用する。

【0141】

該システムは、理想的には、インクジェットプリンターにおけるインクのように、プリンターの消耗品を認証するために使用される。

【0142】

実際に、内蔵型インクジェットプリンターを有するカメラシステムにおいて、消耗品であるインクに関連して、好ましい実施例が論じられるだろうけれども、本発明がそのことに限定されるものではない。

【好適なその他の実施例の説明】

【0143】

好適な実施例により構成されたデジタル画像処理カメラシステムは、図1に示されている。カメラユニット1は、一体的なプリントロール（図示しない）を挿入する手段を含む。カメラユニット1は、カメラによって撮影される像3を検出するエリアイメージセンサ2を含む。オプションとして、シーン3を映像化し、オプション的に立体画法の出力効果の生成物を得るため、第2のエリアイメージセンサを搭載してもよい。

【0144】

カメラ1は、センサ2によって検出された像を表示するためのオプションのカラーディスプレイ5を含む。シンプルな像がディスプレイ5に表示されるとき、ボタン6を押下することが可能であり、その結果、印刷された画像8がカメラユニット1によって出力される。以下では、Artcard（アートカード）9と呼ばれる一連のカードは、一方の面に符号化された情報を含み、もう一方の面に、Artcard9により生成された特定の効果によって歪められた画像を含む。Artcard9は、カメラ1の側面にあるArtcardリーダー10に挿入され、挿入されると、Artcard9の面に現れた歪みと同じ方法で歪められた出力画像8が得られる。したがって、この簡単なユーザインタフェースを使用することにより、特定の効果を生じさせることを望むユーザは、多数のArtcard9のうち一枚をArtcardリーダー10に挿入し、像3の写真を撮影するためボタン19を利用することが可能であり、その結果、対応した歪んだ出力画像8が得られる。

【0145】

カメラユニット1は、カメラユニットの内部プリントロールに残っているプリントアウトの数を含む説明情報を表示するための簡単なLCD出力ディスプレイ15に加えて、多数の他の制御ボタン13、14を含み得る。更に、様々な出力フォーマットがCHPスイッチ17によって制御され得る。

【0146】

次に図2を参照すると、カメラ1の内部ハードウェアの概略図が示されている。内部ハードウェアは、Artcam（アートカム）中央プロセッサユニット（ACP）31を中心としている。

【0147】

Artcam中央プロセッサ31

Artcam中央プロセッサ31は、システムの心臓部を形成する多数の機能を提供する。ACP31は、好ましくは、複雑な高速CMOSシステムオンチップとして実現される。一部のフルカスタム領域を有する標準的なセル設計を利用することが推奨される。0.

10

20

30

40

50

25ミクロンCMOSプロセス上の製作は、合理的に小さいダイ面積と共に、要求される密度及び速度を実現する。

【0148】

ACP31によって与えられる機能には次の機能が含まれる：

1. エリアイメージセンサ2の制御及びデジタル化。ACPの3次元立体視バージョンは、第2のオプションのイメージセンサ4が立体視効果のために設けられた二つのエリアイメージセンサインタフェースを必要とする；
 2. エリアイメージセンサ補償、再フォーマッティング、及び画像強調；
 3. 記憶装置33に対するメモリインタフェース及び管理；
 4. Artcard9からのデータの読み出しのために設けられたArtcardリーダーのリニアイメージセンサ34のインタフェース、制御及びアナログ・デジタル変換； 10
 5. デジタル化された符号化Artcard画像からの未加工Artcardデータの抽出；
 6. Artcardの符号化データのリード・ソロモン誤差検出及び訂正。Artcard9の符号化された面は、Artcard9の画像歪み面に表示された効果を生成するためどのように画像を処理するかに関する情報を含む。この情報は、以下では、「Varkスクリプト」と呼ばれるスクリプトの形式である。Varkスクリプトは、望ましい効果を生成するためACP31内で実行されるインタープリタによって利用される；
 7. Artcard9上のVarkスクリプトの解釈；
 8. Varkスクリプトによって指定されたとおりの画像処理動作の実行； 20
 9. 用紙輸送36、ズームレンズ38、自動焦点39、及びArtcardドライバ37のための様々なモーターの制御；
 10. プリントロール42からの写真8を裁断する裁断機41の動作の裁断機アクチュエータ40の制御；
 11. 印刷用画像データの間調化；
 12. 適当な時におけるプリントヘッド44へのプリントデータの供給；
 13. プリントヘッド44の制御；
 14. プリントヘッド44へのインク圧送の制御；
 15. オプションのフラッシュユニット56の制御；
 16. 方位センサ46、自動焦点47及びArtcard挿入センサ49を含むカメラ 30
- 内の様々なセンサの読み出し及びそれに基づく処置；
17. ユーザインタフェースボタン6、13、14の読み出し及びそれに基づく処置。
 18. 状態ディスプレイ15の制御；
 19. ビューファインダー及びプレビュー画像のカラーディスプレイ5への供給；
 20. 電力管理回路51によるACP電力消費を含むシステム電力消費の制御；
 21. 汎用コンピュータへの外部通信52の提供（パートUSBを使用）；
 22. プリントロール認証チップ53の情報の読み出し及び保存；
 23. カメラ認証チップ54の情報の読み出し及び保存；
 24. テキスト修正用のオプションのミニキーボード57との通信。 40

【0149】

水晶振動子58

水晶振動子58は、システムクロック用の周波数基準として使用される。システムクロックは非常に高いため、ACP31は水晶58から得られた周波数を増大させるため位相ロックループクロック回路を含む。

【0150】

イメージセンシング

エリアイメージセンサ2

エリアイメージセンサ2は、そのレンズを通る像を電気信号へ変換する。エリアイメージセンサは、電荷結合素子（CCD）でもよく、アクティブ画素センサ（APS）CMOSイメージセンサでもよい。現在のところ、利用可能なCCDは、通常、かなり高い画質を 50

有するが、CMOSイメージャの開発は、現在、非常に盛んに行われている。CMOSイメージャは、最終的には、画素数の少ないCCDよりも実質的に安価になり、駆動回路及び信号処理を組み込み得ることが予想される。CMOSイメージャは、12インチウェハーに移行しているCMOS製造技術で製作することが可能である。したがって、CCDとCMOSイメージャの間の製造コストの差は増加し、徐々にCMOSイメージャの方が支持される可能性が高い。しかし、現在のところ、CCDは、最良の選択肢である。

【0151】

Artcamユニットは、1500×1000型のエリアイメージセンサで適切な結果を生じるであろう。しかし、例えば、750×700型のような小型のセンサも、様々な市場で適切であろう。Artcamは、従来のデジタルカメラと比べると、イメージセンサの解像度に対する感度が低い。その理由は、Artcard9に入っている殆どのスタイルは、解像度の不足をわかりにくくするように画像を処理するからである。例えば、画像が印象派の絵画へ変換される効果をシミュレートするため歪められる場合、原画像の低解像度は最小の影響で使用可能である。低解像度入力画像が典型的に気付かれない更なる例には、極めて歪んだ画像を生じる画像ワープ、画像の複数の縮写（例えば、パスポート写真）、ベースレリーフメタルルックのためのバンプマッピングのようなテクスチャ処理、及び構造化されたシーンへの写真合成が含まれる。

【0152】

このように低解像度イメージセンサを許容することは、Artcamユニット1カメラの製造コストを低下させる重要なファクタである。低コストの750×500型イメージセンサを用いるArtcamは、しばしば、非常に高価な1500×1000型イメージセンサを用いる従来のデジタルカメラよりも優れた結果を生じる。

【0153】

オプションの立体視3次元イメージセンサ4

Artcamユニット1の3次元バージョンは、立体視動作のため補助イメージセンサ4を備える。このイメージセンサは、主イメージセンサと同一である。オプションのイメージセンサを駆動する回路は、設計コストの増大を低減するため、ACPチップ31の標準部品として収容してもよい。或いは、別個の3次元Artcam

ACPを設計してもよい。このオプションは、主流の単一センサ型のArtcamの製造コストを削減する。

【0154】

プリントロール認証チップ53

小型チップ53は、各プリントロール42に収容される。このチップは、APS（新写真システム）フィルムカートリッジのような他の形態のカメラフィルムユニット上のバーコード、光学センサ及びホイール、並びに、ISO/ASAセンサの機能を置き換える。

【0155】

認証チップは、以下のその他の機能も提供する：

1. APSロールから機械的及び光学的に検出されるデータではなく、データの保存；
2. 高解像度の正確さの媒体の残りの長さの表示；
3. 粗悪な模造プリントロールコピーを防止するための認証情報。

【0156】

認証チップ53は、1024ビットのフラッシュメモリを含み、1024ビットのうちの128ビットは認証鍵であり、512ビットは認証情報である。更に、認証鍵が直接アクセスされないことを保証するため、暗号化回路が収容される。

【0157】

プリントヘッド44

Artcamユニット1は、十分に小規模、十分に低電力、十分に高速、十分に高品質、及び十分に低コストであり、プリントロールに適合したどのようなカラー印刷技術でも利用可能である。以下、当該印刷ヘッドについて具体的に説明する。

【0158】

インクジェットヘッドの仕様は以下の通りである。

【0159】

【表1】

画像タイプ	2 値、ディザあり
カラー	CMY プロセスカラー
解像度	1600dpi
プリントヘッド長	ページ幅(100mm)
印刷速度	写真 1 枚当たり 2 秒

10

オプションのインク圧コントローラ（図示せず）

インク圧コントローラの機能は、A r t c a mに組み込まれたインクジェットプリントヘッド44のタイプに依存する。一部のタイプのインクジェットでは、インク圧は単純に大気圧であるため、インク圧コントローラを使用しなくても済む。他のタイプのプリントヘッドは、安定化した正のインク圧を必要とする。この場合、インプレッシャ・コントローラは、ポンプ及び圧力トランスデューサを含む。

【0160】

20

その他のプリントヘッドは、インク圧力に、典型的に約100KHzの規則的な振動を生じさせるため超音波トランスデューサが必要である。この場合、A C P 3 1は、これらの振動の周波數位相及び振幅を制御する。

【0161】

用紙輸送モーター36

用紙輸送モーター36は、プリントロール42内からプリントヘッドの傍を通して、一定速度で用紙を移動させる。モーター36は、用紙を移動させるローラーを駆動するため適切な速度までギアダウンされた小型モーターである。機械的なガタ又はその他の振動が印刷されたドット行間隔に影響を与えるので、高画質を実現するためには、高品質モーターと機械式ギアが必要である。

30

【0162】

用紙輸送モータードライバ60

モータードライバ60は、A P C 3 1からのデジタルモーター制御信号をモーター36の駆動に適切なレベルまで増幅する小型回路である。

【0163】

用紙プルセンサ

用紙プルセンサ50は、印刷プロセス中にユーザによるカメラユニットからの写真の引っ張りを検出する。A P C 3 1は、このセンサ50を読み出し、この状況を検出した場合、裁断機41を作動する。用紙プルセンサ50は、動作中にカメラをより絶対確実にするために組み込まれる。ユーザが印刷中に強引に用紙を引き出そうとした場合、プリント機構44又はプリントロール42は、（極端なケースでは）破損するであろう。ポッドが完全に排出される前にポラロイド式カメラからポッドを引き出すことが許されるので、一般の人々はこのような動作を行うことに慣れている。したがって、一般の人々は、用紙を引き出してはならないという印刷された指示に留意しない可能性が高い。

40

【0164】

A r t c a mは、好ましくは、引っ張り検出後に裁断機41が用紙を裁断した後、写真印刷プロセスを再開する。

【0165】

引っ張りセンサは、歪みゲージセンサとして実現してもよく、又は用紙が引っ張られたときに用紙ドライプローラーに発生するトルクによって偏向される小さいプラスチック製の

50

フラグを検出する光学センサとして実現してもよい。後者の実現形態は、低コストのため推奨される。

【0166】

用紙裁断機アクチュエータ40

用紙裁断機アクチュエータ40は、写真の終わりに、又は用紙引っ張りセンサ50が作動されたときに、裁断機41に用紙を裁断させる小型アクチュエータである。

【0167】

用紙裁断機アクチュエータ40は、APCからの裁断機制御信号をアクチュエータ41によって要求されるレベルまで増幅する小型回路である。

【0168】

Artcard9

Artcard9はArtcamユニット用のプログラム記憶媒体である。前述の通り、プログラムは、Varkスクリプトの形式である。Varkは、特に、Artcamユニットのため開発された強力な画像処理言語である。各Artcard9は一つのVarkスクリプトを格納し、これによって、一つの画像処理スタイルを規定する。

【0169】

好ましくは、VARK言語は高度に画像処理専用である。高度に画像処理専用とすることにより、カード上に詳細を記憶するために要する記憶容量は実質的に減少する。更に、機能強化された効果を含む新しいプログラムを作成できる簡便さは、実質的に向上する。好ましくは、この言語は、ワーブマップによる画像ワーピング、コンボリューション、カラーラックアップテーブル、画像のポスタリゼーション、画像へのノイズ付加、画像強調フィルタ、ペインティングアルゴリズム、ブラシジッタリング及び操作エッジ検出フィルタ、タイリング、光源による照明、パンブマップ、テキスト、顔検出及び物体検出属性、3次元フォントを含むフォント、並びに、任意の複合予備レンダリングアイコンを含む多数の画像処理機能を取り扱う機能を含む。Vark言語インタープリタの動作のより詳細な内容は以下に記載される。

【0170】

したがって、作成された言語によって定義されるような言語構造を利用することにより、任意の画像に対する新しい影響を作成し、Artcard上の安価な記憶装置のために構築し、その後、カメラの所有者に配布することが可能である。更に、カードの片面には、カードのもう一方の面に記憶された特定のVARKスクリプトが任意の撮影像に与える影響を説明する例が設けられる。

【0171】

このようなシステムを利用することにより、VARKインタープリタがカメラ装置に組み込まれているならば、装置に依存しないシナリオが与えられ、その結果として、根本的な技術が長期のうちに完全に変えられるという点で、カメラ技術は、著しく陳腐化する虞を伴わずに普及させることができる。更に、新しいフィルタが作成され、カード読み出し用の簡単なカードを用いるような、安価な方式で配布されたときに、Varkスクリプトを更新することが可能である。

【0172】

Artcard9は、クレジットカード（長さ86mm、幅54mm）と同一フォーマットの薄い白色プラスチック製の片である。Artcardは、高解像度インクジェットプリンタを使用して両面印刷されている。このインクジェットプリンタ技術は、1600dpi（63dppmm）の解像度のArtcamに使用される技術と同じ技術であると考えられる。Artcard9の主要な特徴は、安価な製造コストである。Artcardは、プラスチックフィルムの幅広いウェブと同様に高速で製造可能である。プラスチックウェブは、親水性の染料固着層で両面を覆われる。ウェブは、ページ幅のカラーインクジェットプリンタを用いて両面同時に印刷される。次に、ウェブは、個別のカードに切断され打ち抜かれる。カードの一方の面は、Artcard9が検出された像に与える影響を人が読める形で表現したものが印刷される。これは、カードの裏面に記憶されたVarkス

10

20

30

40

50

クリプトを使用して処理された単純な標準画像でもよい。

【0173】

カードの裏面には、画像処理シーケンスを定義する V a r k スクリプトに復号化され得るドットの配列が印刷される。この印刷面積は 80 mm × 50 mm であり、全部で 15876000 ドットがある。このドットの配列は、少なくとも 1.89 M B y t e のデータを表現することができる。高信頼性を実現するため、広範な誤り検出及び訂正がドットの配列に組み込まれる。これにより、カードの実質的な部分が汚損され、損耗され、皺を付けられ、或いは、汚されても、データの完全性に影響が及ばないようにすることができる。使用されるデータ符号化方式はリード・ソロモン符号化であり、データの半分が誤り訂正のために用いられる。これにより、各 A r t c a r d 9 には、誤り訂正された 967 K b y t e のデータを記憶させることができる。

10

【0174】

リニアイメージセンサ 34

A r t c a r d リニアセンサ 34 は、上記の A r t c a r d データ画像を電気信号へ変換する。エリアイメージセンサ 2、4 の場合と同様に、リニアイメージセンサは、C C D 又は A P S

C M O S 技術を用いて製造可能である。イメージセンサ 34 の有効長さは 50 mm であり、A r t c a r d 9 上のデータ配列の幅と一致する。ナイキストのサンプリング定理を充たすため、リニアイメージセンサ 34 の解像度は、イメージセンサに到達する A r t c a r d 光学像の最高空間周波数の少なくとも 2 倍でなければならない。實際上、データ検出は、イメージセンサ解像度がこれよりも実質的に高くなった場合、より簡単になる。4900 d p i (180 d p m m) の解像度が選ばれた場合、全部で 9450 画素が与えられる。この解像度は、5.3 μ m の画素センサピッチを必要とする。これは、千鳥状にされた 4 行の 20 μ m 画素センサを使用することによって簡単に実現することができる。

20

【0175】

リニアイメージセンサは、ライトパイプ（図示せず）によって A r t c a r d 9 を照明するための L E D 65 を含む特殊なパッケージに装着される。

【0176】

A r t c a r d リーダーのライトパイプは、以下の数機能を備えた成形ライトパイプでもよい：

30

1. ライトパイプは、全ての内部反射ファセットを使用してカードの幅に亘って L E D からの光を拡散する。
2. ライトパイプは、集積化されたシリンドリカルレンズを使用して A r t c a r d 9 の 16 μ m 幅のストリップ上に光を集める。
3. ライトパイプは、成形されたマイクロレンズの配列を使用して A r t c a r d から反射された光をリニアイメージセンサの画素に集める。

【0177】

次に、A r t c a r d リーダーの動作を説明する。

【0178】

A r t c a r d リーダーモーター 37

40

A r t c a r d リーダーモーターは、比較的一定の速度で A r t c a r d を進めてリニアイメージセンサ 34 を通過させる。厳密な精度の機械コンポーネントを A r t c a r d リーダーに収容することは費用対効果がよくないので、モーター 37 は、A r t c a r d 9 を動かすローラーのペアを駆動するため適切な速度までギアダウンされた標準的な小型モーターである。速度変動、ガタ、及びその他の振動は、未加工の画像データに影響を与えるので、A P C 31 内の回路は、A r t c a r d データを確実に読むためこれらの効果の広範な補償を行う。

【0179】

モーター 37 は、A r t c a r d を取り出すとき、逆向きに駆動される。

【0180】

50

Art card モータードライバ 6 1

Art card モータードライバ 6 1 は、モーター 3 7 を駆動するために適切なレベルまで APC 3 1 からのデジタルモーター制御信号を増幅する小型回路である。

【0181】

カード挿入センサ 4 9

カード挿入センサ 4 9 は、カードリーダー 3 4 に挿入されているカードの有無を検出する光学センサである。このセンサ 4 9 からの信号の後、APC 3 1 はカード読み出しプロセスを開始し、そのプロセスには、Art card リーダーモーター 3 7 の作動が含まれる。

【0182】

カードイジェクトボタン 1 6

カードイジェクトボタン 1 6 (図 1) は、現在の Art card を取り出すためユーザによって使用され、他の Art card を挿入できるようになる。APC 3 1 は、ボタンの押下を検出し、カードを取り出すため、Art card リーダーモーター 3 7 を反転させる。

【0183】

カード状態インジケータ 6 6

カード状態インジケータ 6 6 は、Art card 読み出しプロセスの状態をユーザへ報せるために設けられる。これは、標準的な 2 色 (赤/緑) LED でもよい。カードが正しく読み出され、データの完全性が検証されたとき、LED は連続的に緑色で発光する。カードが不良である場合、LED は赤色で発光する。

【0184】

カメラが 3 V のバッテリーではなく 1. 5 V のバッテリーから給電される場合、電源電圧は、緑色 LED の前向き電圧降下よりも小さく、LED は発光しない。この場合、赤色 LED が使用されるか、又はその LED は、より高い電圧を必要とする Art cam 内の他の回路に給電する電圧ポンプから給電される。

【0185】

6 4 M ビット DRAM 3 3

多種多様な画像処理効果を実行するため、カメラは 8 M b y t e のメモリ 3 3 を利用する。これは、単一の 6 4 M ビットメモリチップによって設けることができる。勿論、メモリ技術の変化に伴って、より大容量の DRAM 記憶容量サイズで置き換えても構わない。

【0186】

メモリチップへの高速アクセスが要求される。これは、R a m b u s

DRAM (毎秒 5 0 0 メガバイトのバーストアクセス速度)、又はダブルデータレート (DDR) SDRAM 又はシンクリンク (S y n c L i n k) DRAM チップを使用することによって達成できる。

【0187】

カメラ認証チップ

カメラ認証チップ 5 4 は、異なる情報を記憶している点を除くと、プリントロール認証チップ 5 3 と同一のものである。カメラ認証チップ 5 4 には、以下の三つの主要な目的がある：

1. 認証コードをプリントロール認証チップと比較するセキュア手段を設けること；
2. カメラの製造番号のような製造情報用の記憶装置を設けること；
3. ユーザ情報の記憶用の小容量の不揮発性メモリを設けること。

【0188】

ディスプレイ

Art cam は、オプションのカラーディスプレイ 5 及び小型状態ディスプレイ 1 5 を含む。最低コストの民生用カメラは、ある種のデジタルカメラ及びビデオカメラで見られるような小型 TFT

LCD 5 のようなカラー画像ディスプレイを含む。カラーディスプレイ 5 は、これらのバ

10

20

30

40

50

ージョンの A r t c a m の主要なコストのかかる要素であり、バックライトを加えたディスプレイ 5 は主要な電力消費元である。

【0189】

状態ディスプレイ 1 5

状態ディスプレイ 1 5 は、小型の受動セグメントに基づく L C D であり、現在のハロゲン化銀式カメラ及びデジタルカメラに設けられている L C D と類似している。その主要な機能は、プリントロール 4 2 に残っているプリントの数と、フラッシュ状態及びバッテリー状態のような様々な標準的なカメラ機能に対するアイコンと、を表示することである。

【0190】

カラーディスプレイ 5

カラーディスプレイは、完全動画像ディスプレイであり、ビューファインダーとして、印刷される画像の確認として、及びユーザインタフェースディスプレイとして動作する。ディスプレイ 5 のコストは、その面積にほぼ比例するので、大型ディスプレイ（例えば、4 インチ対角）ユニットは、A r t c a m ユニットの高価なバージョンに限定される。例えば、カラービデオカメラの約 1 インチのビューファインダー T F T のようなより小さいディスプレイは、中程度の A r t c a m に有効である。

【0191】

ズームレンズ（図示せず）

A r t c a m はズームレンズを含むことができる。これは、標準的な電子カメラに使用されるズームレンズと同一であり、ポケットカメラのズームレンズに類似している標準的な電子制御ズームレンズでもよい。A r t c a m ユニットの好適なバージョンは、標準的な交換可能な 3 5 m m S L R レンズを含む。

【0192】

自動焦点モーター 3 9

自動焦点モーター 3 9 はズームレンズの焦点を変更する。モーターは、自動焦点機構を駆動するため適切な速度にギアダウンされた小型モーターである。

【0193】

自動焦点モータードライバ 6 3

自動焦点モータードライバ 6 3 は、モーター 3 9 を駆動するため適切なレベルまで A P C 3 1 からのデジタルモーター制御信号を増幅する小型回路である。

【0194】

ズームモーター 3 8

ズームモーター 3 8 は、ズームのフロントレンズを出入りするように動かす。モーターは、ズーム機構を駆動するため適切な速度にギアダウンされた小型モーターである。

【0195】

ズームモータードライバ 6 2

ズームモータードライバ 6 2 は、モーターを駆動するため適切なレベルまで A P C 3 1 からのデジタルモーター制御信号を増幅する小型回路である。

【0196】

通信

A C P 3 1 は、パーソナルコンピュータとの通信用のユニバーサルシリアルバス（U S B ）インタフェース 5 2 を含む。必ずしも全ての A r t c a m モデルが U S B コネクタを組み込むことを予定されているわけではない。しかし、U S B 回路 5 2 のために必要なシリコン面積は小さいので、インタフェースを標準的な A C P に収容することが可能である。

【0197】

オプションのキーボード 5 7

A r t c a m ユニットは、A r t c a r d によって指定されたテキストをカスタマイズするオプションの小型キーボード 5 7 を含む。A r t c a r d 画像に現れるあらゆるテキストは、複雑なメタリック 3 次元フォントの形である場合でも編集可能である。小型キーボードは、元のテキストと編集されたテキストを表示するため単一ラインの英数字 L C D を

10

20

30

40

50

含む。このキーボードは標準装備品でもよい。

【0198】

ACP31は、小型キーボードとの間でデータを転送するシリアル通信回路を含む。

【0199】

電源

Artcamユニットは、バッテリー48を使用する。Artcamのオプションに依存して、このバッテリーは、3Vのリチウム電池、1.5VのAAアルカリ電池、又はその他のバッテリー装置のいずれでもよい。

【0200】

電源管理ユニット51

電源消費は、Artcamにおける重要な設計制約条件である。標準的なカメラバッテリー（例えば、3Vのリチウムバッテリー）又は標準的なAA若しくはAAAアルカリ電池の何れかが使用可能であることが望ましい。Artcamユニットの電子回路の複雑さの程度は35mm写真カメラよりも著しく高いが、電力消費量が相応に高いとは限らない。Artcamの電源は、全てのユニットが未使用時には電源をオフされるように慎重に管理される。

【0201】

最も重大な電力浪費元は、ACP31、エリアイメージセンサ2及び4、プリンタ44の様々なモーター、フラッシュユニット56、並びに、オプションのカラーディスプレイ5である。次に各部品について別々に取り扱う：

1. ACP：0.25 μ mのCMOSを使用して製造され、1.5Vで動作するならば、ACP電力消費は非常に小さくすることができる。ACPチップの様々な部品へのクロックも非常に小さくすることができる。ACPチップの様々な部品へのクロックは、未使用時にオフにすることができ、スタンバイ電流消費を殆ど取り除く。ACPは、印刷される写真毎に約4秒間だけ完全に使用される；

2. エリアイメージセンサ：電力は、ユーザがボタンに指を置いたときに限りエリアイメージセンサに供給される；

3. プリンタ電力は、実際に印刷しているときに限りプリンタへ供給される。これは、1枚の写真につき約2秒間である。たとえそうであっても、適度に低い電力消費印刷が使用されるべきである；

4. Artcamで必要になるモーターは、全て低電力小型モーターであり、典型的に、1枚の写真について数秒間だけ作動される；

5. フラッシュユニット45は、一部の写真だけに対して使用される。その電力消費は、妥当なバッテリー寿命の3Vのリチウムバッテリーによって容易に供給される。

6. オプションのカラーディスプレイ5は、次の二つの理由から主要な電流浪費元である。カラーディスプレイは、カメラの使用中の全ての時間に亘ってオンにする必要がある、液晶ディスプレイが使用された場合には、バックライトが必要になる。カラーディスプレイを組み込むカメラは、許容可能なバッテリー寿命を達成するため、より大型のバッテリーを要求する。

【0202】

フラッシュユニット56

フラッシュユニット56は民生用カメラの標準的な小型電子フラッシュである。

【0203】

ACP31の概要

図3は、Artcam中央プロセッサ（ACP）31を詳細に示す図である。Artcam中央プロセッサは、Artcamの処理能力の全てを提供する。ACPは、0.25ミクロンCMOSプロセス用に設計され、約150万個のトランジスタと、約50mm²の面積を有する。ACP31の設計は複雑であるが、設計の手間は、データパスコンパイレーション技術、マクロセル、及びIPコアを使用することによって軽減できる。ACP31は、

10

20

30

40

50

RISC型CPUコア72と、
 4ウェイVLWベクトルプロセッサ74と、
 ダイレクトRAMbus（ラムバス）インタフェース81と、
 CMOSイメージセンサインタフェース83と、
 CMOSリニアイメージセンサインタフェース88と、
 USBシリアルインタフェース52と、
 赤外線キーボードインタフェース55と、
 数字LCDインタフェース84と、
 カラーTFT LCDインタフェース88と、
 プログラム記憶装置70用4Mバイトフラッシュメモリ70と、
 を含む。RISC型CPU、ダイレクRAMbusインタフェース81、CMOSセンサ
 インタフェース83、及びUSBシリアルインタフェース52は、ベンダーによって供給
 されるコアでもよい。ACP31は、電力消費を最小限に抑えるため、外部3V及び内部
 1.5Vに基づいて200MHzのクロック速度で動作することが意図されている。CPU
 コアは100MHzで動作すればよい。以下の二つのブロック図、即ち、
 単独のACP31の説明図、及び
 Artcamの残りのハードウェアに接続されたACP31の上位レベルの図を与える例
 示的なArtcam、
 はACP31の2通りの見方を与える。

10

20

【0204】

画像アクセス

前述の通り、DRAMインタフェース81は、ACPチップの他のクライアント部と、ラ
 ムバスDRAMとの間を接続する役割を担う。實際上、DRAMインタフェース内の各モ
 ジュールはアドレス生成器である。

【0205】

ACPによって操作される画像には三つの論理タイプがある。それらは、
 - CCDから撮影された入力像であるCCD画像と、
 - Artcam装置によって内部的に利用される画像フォーマットである内部画像フォー
 マットと、
 - Artcamによって印刷される出力画像フォーマットである印刷画像と、
 である。

30

【0206】

これらの画像は、典型的に、色空間、解像度、並びに、カメラ間で変動のあり得る出力及
 び入力色空間の点で相違する。例えば、低価格カメラのCCD画像は、高価格カメラで使
 用されるものとは異なる解像度、又は異なる色特性を揺する。しかし、あらゆる内部画像
 フォーマットは、全てのカメラを通じて色空間の点で同一フォーマットである。

【0207】

その上、3種類の画像タイプは、どちらの方向が「上向き」であるかに関しても異なり得
 る。カメラの物理的な向きは、人物像であるか、又は風景像であるかについての観念を生
 じさせ、この観念は処理全体を通じて維持されるべきである。このため、内部画像は常に
 正しく向きを決められ、印刷動作中にCCDから獲得された画像に対して回転が実行され
 る。

40

【0208】

CPUコア（CPU）72

ACP31は、Vark画像処理言語インタープリタを動作させ、Artcamの汎用オ
 ペレーティングシステムの役割を実行するため32ビットRISC型CPU72を内蔵す
 る。多種多様なCPUコアが適当であり、要求されたコア計算及び制御機能を消費者の期
 待を満足する十分な速さで実行するために足る処理能力を備えていればどのようなプロ
 セッサコアでも構わない。適当なコアの例には、LSIロジック社のMIPS
 R4000、Strong ARMコアが含まれる。異なるArtcamモデルの間で命

50

令セットの継続性を維持する必要はない。A r t c a r dの互換性は、将来のプロセッサの進歩及び変更とは無関係に維持される。なぜならば、V a r kインタープリタは、新しい命令セット毎に再コンパイルをするだけで済むからである。したがって、A C P 3 1のアーキテクチャは、自由に進化させることができる。種々の製造者は、C P Uコアの使用許諾や移植をすることなく、様々なA C P 3 1チップ設計を製造することができる。この装置非依存性によって、P C市場においてインテル社が行っているようなチップベンダーのロックインが避けられる。C P Uは1 0 0 M H zで動作し、単一サイクルタイムは1 0 n sである。このC P Uは、V L I Wベクトルプロセッサ7 4が時間的に重大な演算を担うとしても、V a r kインタープリタを実行するために十分な高速性が必要である。

【0209】

プログラムキャッシュ72

プログラムコードがオンチップフラッシュメモリ70に保存されとしても、ウェルパックされたフラッシュメモリ70がC P Uによって要求される1 0 n sのサイクルタイムで動作できる可能性は低い。したがって、優れた性能のために小規模のキャッシュが要求される。各々が32バイトの16個のキャッシュラインは、全部で512バイトになり、十分な大きさである。プログラムキャッシュ72は、プログラムキャッシュ72という名前の章に規定されている。

【0210】

データキャッシュ76

小規模のデータキャッシュ76が優れた性能のために要求される。この要求は、概して、ラムバスD R A Mを使用しているために生じる。ラムバスD R A Mは、高速データをバーストで提供することができるが、単一バイトのアクセスには不十分である。C P Uは、C P Uデータキャッシュ76のサイズをフレキシブルに操作することを可能にさせるメモリキャッシングシステムにアクセスする。最小限で16個のキャッシュライン(512バイト)が優れた性能のために推奨される。

【0211】

C P Uメモリモデル

A r t c a mのC P Uメモリモデルは、32MBのエリアにより構成される。A r t c a mの基本モデルでは、C P Uメモリモデルは、8MBの物理的R D R A Mオフチップにより構成され、最大で16MBのオフチップメモリが設けられる。A C P 3 1にはプログラム記憶用に4MBのフラッシュメモリ70が存在し、最終的に、4MBのアドレス空間がA C P 3 1の様々なレジスタ及びコントロールに割り当てられる。したがって、A r t c a m用のメモリマップは以下の通りである。

【0212】

【表2】

内 容	サイズ
基本 Artcam DRAM	8MB
拡張 DRAM	8MB
プログラムメモリ(ACP31 上のフラッシュメモリ 70 内)	4MB
プログラムメモリの拡張用予備	4MB
ACP31 レジスタ及びメモリマップド I/O	4MB
予 備	4MB
合 計	32MB

アドレスをデコードする簡単な方法は、アドレスビット23-24を使用することである。即ち、

ビット24がクリアされている場合、アドレスは下位側16MBのレンジに含まれるので、D R A M及びデータキャッシュ76から充たされる。殆どの場合に、D R A Mは8MB

10

20

30

40

50

に過ぎないが、より上位のメモリモデル *Artcam* を考慮に入れるため 16 MB が割り当てられる。

ビット 24 がセットされ、ビット 23 がクリアされている場合、アドレスは、フラッシュメモリ 70 の 4 M バイトレンジを表現し、プログラムキャッシュ 72 によって充たされる。

ビット 24 = 1 及びビット 23 = 1 である場合、アドレスは、CPU メモリデコーダ 68 によって、低速バス経由による AC 内の要求されたコンポーネントへのアクセスに変換される。

【0213】

フラッシュメモリ 70

ACP 31 は、*Artcam* プログラムを記憶する 4 M バイトのフラッシュメモリ 70 を含む。フラッシュメモリ 70 はマスク ROM よりも高い充填率を有し、カメラプログラムコードを試験するためより高いフレキシビリティを許容することが想定される。フラッシュメモリ 70 の弱点はアクセス時間であり、このアクセス時間は CPU の 100 MHz の動作速度 (10 ns のサイクルタイム) に対して十分に高速であるとはいえない。したがって、高速プログラム命令キャッシュ 77 が、CPU と低速のフラッシュメモリ 70 との間のインタフェースとして作用する。

【0214】

プログラムキャッシュ 72

優れた CPU 性能のために小型キャッシュが要求される。この要求は、プログラムコードを記憶するフラッシュメモリ 70 が低速であることに起因する。各々が 32 バイトの 16 個のキャッシュラインは、全部で 512 バイトになり、十分な大きさである。プログラムキャッシュ 72 は読み出し専用キャッシュである。CPU プログラムによって使用されるデータは、CPU メモリデコーダ 68 を通過し、アドレスが DRAM 内にある場合、汎用データキャッシュ 76 を通る。分離によって、CPU を VLIW ベクトルプロセッサ 74 とは独立して動作させることができる。所定のプロセスに対するデータ要求量が少ない場合、結果的に、そのプロセスは完全にキャッシュによって動作し得る。

【0215】

最後に、プログラムキャッシュ 72 は、CPU によって純粹にプログラム命令としてではなく、データとして読み出すことが可能である。これにより、テーブル、即ち、VLIW のためのマイクロコード等をフラッシュメモリ 70 からロードすることが可能になる。ビット 24 がセットされ、ビット 23 がクリアされているアドレスは、プログラムキャッシュ 72 から充足される。

【0216】

CPU メモリデコーダ 68

CPU メモリデコーダ 68 は、CPU データアクセスに沿う簡単なデコーダである。デコーダは、データアドレスを、内部低速バスによる内部 ACP レジスタアクセスに変換し、これにより、ACP レジスタのメモリマップド I/O を可能にさせる。CPU メモリデコーダ 68 は、ビット 24 がセットされ、ビット 23 がクリアされているアドレスだけを解釈する。CPU メモリデコーダ 68 内ではキャッシングは行われない。

【0217】

DRAM インタフェース 81

Artcam によって使用される DRAM は、1.6 GB/秒で動作する単一チャンネルの 64 M ビット (8 M バイト) RAMbus (ラムバス) RDRAM である。RDRAM アクセスは、単一チャンネル (16 ビットデータバス) コントローラによる。RDRAM は、低電力動作の数種の有効な動作モードを有する。ラムバス仕様書には、95% 以上の効率を達成可能であるようなランダムな 32 バイト転送のあるシステムが記載されているが、これは、32 バイトのうちの一部しか使用されない場合には成り立たない。同一のデバイスに対する 2 回の読み出しに続いて 2 回の書き込みを行うことによって 86% 以上の効率が得られる。基礎的なレイテンシーは、書き込みから読み出しへ変化するバスのタ

10

20

30

40

50

ーンアラウンドのために必要であり、遅延書き込み機構が存在するので、効率を更に改善することが可能である。書き込みに関して、書き込みマスクは、特定のバイトのサブセットへの書き込みを可能にさせる。これらの書き込みマスクは、内部キャッシュの「ダーティビット」によってセットされるであろう。ラムバスダイレクト R D R A M は、要するに、1 G B / 秒を超えるスループットを簡単に達成可能であり、32 バイトの転送知識を十分に利用する知的アルゴリズムと組み合わせられ（殆どのプロセスで）書き込み毎に複数回の読み出しを伴い、1.3 G B / 秒を超える転送レートが期待されるということである。10 n s 毎に、コアとの間で16 バイトを転送することができる。

【0218】

D R A M 構成

基本モデル（8 M B R D R A M）の A r t c a m 用の D R A M 構成を次に示す。

【0219】

【表3】

内 容	サイズ
プログラムスクラッチ RAM	0.50MB
Artcard データ	1.00MB
CMOS センサから撮影された写真画像	0.50MB
印刷画像(圧縮形式)	2.25MB
拡張写真画像の1チャンネル	1.50MB
単一チャンネルの一つの画像ピラミッド	1.00MB
中間画像処理	1.25MB
合 計	8MB

注：非圧縮形式の印刷画像は4.5 M B（1チャンネル当たり1.5 M B）を必要とする。他の対象を8 M B モデルに収容するため、印刷画像は圧縮されるべきである。クロミナンスチャンネルが4：1に圧縮された場合、各クロミナンスチャンネルは0.375 M B しか必要としない。

【0220】

ここで説明されるメモリモデルは、単一の8 M B の R D R A M を想定している。A r t c a m のその他のモデルの中にはより多くのメモリを具備し、印刷画像を圧縮しなくてもよいものがある。その上、メモリ量が増加すると、最終画像の大部分を同時に操作できるようになり、速度が向上する可能性がある。

【0221】

A r t c a r d の取り出し又は挿入は、印刷画像、拡大された写真画像の1チャンネル、及び画像ピラミッドを保持する5.5 M B エリアを無効にすることに注意すべきである。この空間は、A r t c a r d データを復号化するため A r t c a r d インタフェースによって安全に使用される。

【0222】

データキャッシュ76

A C P 3 1 は、専用 C P U 命令キャッシュ77及び汎用データキャッシュ76を含む。データキャッシュ76は、C P U、V L I W ベクトルプロセッサ74及びディスプレイコントローラ88からの全ての D R A M 要求（データの読み出し及び書き込み）を取り扱う。これらの要求は、メモリ使用量及びアルゴリズム的タイミング必要条件の点で非常に異なる側面を有する。例えば、V L I W プロセスは、リニアメモリ内の画像を処理し、画像内の値毎にテーブルの値をルックアップする。画像の大半をキャッシュする必要な殆ど無いが、実際のメモリアクセスが必要ではないように、ルックアップテーブル全体をキャッシュすることが望ましい。このように必要条件が相違しているため、データキャッシュ76は、キャッシングを知的に定義することが可能である。

【0223】

ラムバスDRAMインタフェース81は、非常に高速のメモリアクセス能力を備えているが（平均スループットは25nsで32バイト）、単一バイトの要求を処理するためには不十分である。実効的なメモレイテンシーを低下させるため、ACP31は、128個のキャッシュラインを含む。各キャッシュラインは32バイト幅である。このようにして、データキャッシュ76の総容量は4096バイト（4KB）である。128個のキャッシュラインは、16個のプログラム可能なサイズのグループに構成される。16個のグループの各々は、連続したキャッシュラインのセットでなければならない。CPUは、各グループに割り付けるキャッシュラインの個数を決定する役割を担う。各グループ内で、キャッシュラインは、簡単なLRU（最低使用頻度）アルゴリズムに従って詰められる。CPUデータ要求に関して、データキャッシュ76は、アドレスビット24がクリアされたメモリアクセス要求を取り扱う。ビット24がクリアされている場合、アドレスは下位側16MBのレンジに収まるので、DRAM及びデータキャッシュ76から応じることができる。殆どの場合に、DRAMは、8MBしかないが、よりメモリ量の多いモデルのArtcamを考慮に入れて16MBが割り付けられる。ビット24がセットされている場合、アドレスはデータキャッシュ76によって無視される。

10

【0224】

全てのCPUデータ要求はキャッシュグループ0から応じられる。優れたCPU性能のためには16個の最小のキャッシュラインが推奨されるが、CPUは、（0個以外の）任意の個数のキャッシュラインをキャッシュグループ0に割り当てることができる。残りのキャッシュグループ（1から15）は、現在の必要条件に従って割り付けられる。これは、VLIWベクトルプロセッサ74プログラム又はディスプレイコントローラ88への割付を意味することもある。例えば、常時利用可能であることが要求される256バイトのルックアップテーブルは、8個のキャッシュラインが必要であろう。順次画像を書き出すためには、（生成されたレコードのサイズと、書き込み要求が多数のサイクルに亘って書き込み遅延されているかどうかと、に応じて）2から4個のキャッシュラインだけが必要であろう。各キャッシュラインのバイトには、メモリをDRAMへ書き込む際に書き込みマスクを作成するため使用されるダーティビットが関連付けられる。各キャッシュラインには別のダーティビットが関連付けられ、このダーティビットは、キャッシュラインのバイトの何れかに書き込みがされたかどうか（したがって、DRAMを再使用する前に、キャッシュラインをDRAMへ書き戻さなければならないかどうか）を示す。二つの異なるキャッシュグループがメモリの同じアドレスをアクセスし、同期が外れる可能性があることに注意する必要がある。VLIWプログラムライターは、これが問題にならないことを保証する役割を担う。例えば、あるキャッシュグループには画像を読み出す役割を与え、別のキャッシュグループには変更された画像をメモリに書き戻す役割を与えることは完全に合理的である。画像が順番に読み出し又は書き込みされる場合、キャッシュラインをこのようにして割り付けることは有利である。全部で8本のバス182は、VLIWベクトルプロセッサ74をデータキャッシュ76へ接続する。各バスは、I/Oアドレス生成器に接続される。（1個の処理ユニット178あたりに2個のI/Oアドレス生成器189及び190が存在し、VLIWベクトルプロセッサ74には4個の処理ユニットが存在する。したがって、バスの総数は8本である。）

20

30

40

所与のサイクルにおいて、CPUのキャッシュグループ（グループ0）への単一の32ビット（4バイト）アクセスの他に、残りのキャッシュグループへの16ビット（2バイト）の四つ同時のアクセスが8本のVLIWプロセッサ74のバス上で許容される。データキャッシュ76は要求を公平に処理する役割がある。ある所与のサイクルで、特定のキャッシュグループへのたった一つの要求が処理される。VLIWベクトルプロセッサ74に8個のアドレス生成器189、190が存在する場合、アドレス生成器の各々は個別のキャッシュグループを参照する可能性がある。しかし、2個以上のアドレス生成器189、190が同じキャッシュグループにアクセスする可能性があり、場合によっては、その方が合理的である。CPUは、キャッシュグループに正確なキャッシュラインの個数が割り

50

付けられることを保証し、V L I Wベクトルプロセッサ74の様々なアドレス生成器189、190が特定のキャッシュグループを正確に参照することを保証する役割を担う。上述のようにデータキャッシュ76は、ディスプレイコントローラ88及びV L I Wベクトルプロセッサ74が同時に動作状態になることを許容する。これらの二つのコンポーネントの動作が決して同時に出現しないと考えられるならば、全部で9個のキャッシュグループがあれば足りる。C P Uはキャッシュグループ0を使用し、V L I Wベクトルプロセッサ74及びディスプレイコントローラ88は残りの8個のキャッシュグループを共用し、(4ビットではなく)3ビットだけで特定の要求に応じるキャッシュグループを定めることができる。

【0225】

10

J T A Gインタフェース85

標準的なJ T A G (ジョイントテスト部会) インタフェースはテスト目的用にA C P 3 1に収容される。チップの複雑さのため、B I S T (ビルトインセルフテスト) 及び機能的ブロック分離等を含む様々なテスト技術が必要である。チップ面積の10%のオーバーヘッドが全チップテスト回路のため想定される。テスト回路は本文書の範囲外の事項である。

【0226】

シリアルインタフェース

U S Bシリアルポートインタフェース52

これは標準的なU S Bシリアルポートであり、内部チップの低速バスに接続されるので、C P Uがそれを制御することができる。

20

【0227】

キーボードインタフェース65

これは標準的な低速シリアルポートであり、内部チップ低速バスに接続されるので、C P Uがそれを制御することができる。キーボードインタフェースは、プリントをカスタマイズするための簡単なデータ入力を許容するためオプション的にキーボードへ接続できるように設計される。

【0228】

認証チップシリアルインタフェース64

これらは、二つの標準的な低速シリアルポートであり、内部チップ低速バスに接続されるので、C P Uがこれらを制御できる。二つのポートを具備する理由は、別個のラインを使用して、オンカメラ認証チップとプリントロール認証チップの両方に接続するためである。1個のラインしか使用しない場合、模造プリントロールの製造者は、認証コードを生成するのではなく、カメラの認証チップによって生成されたコードを使用するようにカメラを誤動作させるチップを設計することができる。

30

【0229】

パラレルインタフェース67

パラレルインタフェースはA C P 3 1を個別の静的な電気信号に接続する。C P Uは、低速バスを介して、メモリマップドI / Oとしてこれらの配線の各々を制御することができる。以下の表は、パラレルインタフェースへの配線の一覧である。

40

【0230】

【表 4】

配 線	向 き	ピン数
用紙輸送ステッピングモーター	出 力	4
Artcard ステッピングモーター	出 力	4
ズームステッピングモーター	出 力	4
裁断機モーター	出 力	1
フラッシュトリガー	出 力	1
状態 LCD セグメントドライバ	出 力	7
状態 LCD 共通ドライバ	出 力	4
Artcard 照明 LED	出 力	1
Artcard 状態 LED(赤／緑)	入 力	2
Artcard センサ	入 力	1
用紙引っ張りセンサ	入 力	1
方位センサ	入 力	2
ボタン	入 力	4
	合 計	36

10

20

V L I W入力及び出力 F I F O 7 8、7 9

V L I W入出力 F I F Oは、プロセスと V L I Wベクトルプロセッサ 7 4との間を接続するため使用される 8ビット幅の F I F Oである。両方の F I F Oは V L I Wベクトルプロセッサ 7 4によって制御されるが、C P Uによってクリアしたり、（例えば、状態について）問い合わせをしたりすることができる。

【 0 2 3 1 】

V L I W入力 F I F O 7 8

クライアントは、V L I Wベクトルプロセッサ 7 4にデータを処理させるため、8ビットデータを V L I W入力 F I F O 7 8に書き込む。クライアントには、イメージセンサインタフェース、A r t c a r dインタフェース、及び C P Uが含まれる。これらのプロセスの各々は、単にデータを F I F Oに書き込み、V L I Wベクトルプロセッサ 7 4に重い仕事の全ての実行を任せることによって、処理から解放される。クライアントによる V L I W入力 F I F O 7 8の使用を用いる一例は、イメージセンサインタフェース（I S I 8 3）である。イメージセンサインタフェース 8 3は、イメージセンサからデータを取得し、そのデータを F I F Oに書き込む。V L I Wプロセスは、F I F Oからデータを取得し、データを正しい画像データフォーマットへ変換し、それを D R A Mへ書き出す。イメージセンサインタフェース 8 3は、その結果として、非常に簡単化される。

30

【 0 2 3 2 】

V L I W出力 F I F O 7 9

V L I Wベクトルプロセッサ 7 4は、8ビットデータを V L I W出力 F I F O 7 9に書き込み、クライアントは V L I W出力 F I F Oでデータを読み出すことができる。クライアントには、プリントヘッドインタフェース及び C P Uが含まれる。これらのクライアントの両方は、単に F I F Oから処理済みのデータを読み出し、V L I Wベクトルプロセッサ 7 4に重い仕事の全ての実行を任せることにより、処理から解放される。C P Uは、データが V L I W出力 F I F O 7 9に収容されたときはいつでも中断させることができ、F I F Oを絶えずポーリングするのではなく、データが利用可能になったときに限りそのデータを処理できるようになる。クライアントによる V L I W出力 F I F O 7 9の使用を用いる一例は、プリントヘッドインタフェース（P H I 6 2）である。V L I Wプロセスは、画像を取得し、画像を正しい向きへ回転し、色変換を行い、プリントヘッド必要条件に応じて得られた画像をディザリングする。プリントヘッドインタフェース 6 2は、V L I W

40

50

出力F I F O 7 9からディザリングされたフォーマット後の8ビットデータを読み出し、その8ビットデータをA C P 3 1の外部にあるプリントヘッドに渡すだけでよい。プリントヘッドインタフェース6 2はその結果として非常に簡単化される。

【0 2 3 3】

V L I Wベクトルプロセッサ7 4

A r t c a mの高い処理要求を実現するため、A C P 3 1は、V L I W（超長命令語）ベクトルプロセッサを含む。V L I Wプロセッサは、クロスバースイッチ1 8 3によって接続され並列に動作する4個の同一処理ユニット（例えば、P U 1 7 8）の組である。各処理ユニット、例えば、P U 1 7 8は、各サイクルに、4回の8ビット乗算、8回の8ビット加算、3回の3 2ビット加算、I / O処理、及び様々な論理演算を実行することができる。処理ユニット、例えば、P U 1 7 8はマイクロコード化され、各々は、2個のアドレス生成器1 8 9、1 9 0を具備し、データ処理のため利用可能なサイクルを完全に使用できる。4個の処理ユニット、例えば、P U 1 7 8は、通常、密接に相互作用するV L I Wプロセッサを実現するため同期させられる。2 0 0 M H zのクロックにより、V L I Wベクトルプロセッサ7 4は、1 2 G o p（每秒1 2 0億回の演算速度）で動作する。命令は、ワーピング、芸術的なブラッシング、複雑な合成照明、色変換、画像フィルタリング、及び合成のような画像処理機能のために調整される。これらは、デスクトップコンピュータよりも2桁加速されている。

10

【0 2 3 4】

図3（a）に詳細に示されるように、V L I Wベクトルプロセッサ7 4は、クロスバースイッチ1 8 3によって接続された4個の処理ユニット、例えば、P U 1 7 8であり、各処理ユニット、例えば、P U 1 7 8は、クロスバースイッチ1 8 3へ2入力进行供給し、クロスバースイッチ1 8 3からの2出力を取得する。2個の共通レジスタは、処理ユニット、例えば、P U 1 7 8のための制御及び同期機構を形成する。8本のキャッシュバス1 8 2は、データキャッシュ7 6を介してD R A Mへの接続を可能にさせ、2本のバスは各処理ユニット、例えば、P U 1 7 8へ至る（I / Oアドレス生成器毎に1本のバス）。各処理ユニット、例えば、P U 1 7 8は、（データを処理するための多数のレジスタ及びある種の算術論理を格納する）A L U（算術論理ユニット）1 8 8と、ある種のマイクロコードR A M 1 9 6と、外界（他のA L Uを含む）への配線と、を具備する。ローカルP U状態機械は、マイクロコードで動き、この手段によって処理ユニット、例えば、P U 1 7 8が制御される。各処理ユニット、例えば、P U 1 7 8は、D R A M（データキャッシュ7 6経由）とA L U 1 8 8（入力F I F O及び出力F I F O経由）の間のデータフローを制御する二つのI / Oアドレス生成器1 8 9、1 9 0を含む。アドレス生成器は、データ（特に、様々なフォーマットの画像）、並びに、D R A M内のテーブル及びシミュレーティッドF I F Oを読み書きすることが可能である。フォーマットはソフトウェア制御の下でカスタマイズ可能であるが、マイクロコード化されない。データキャッシュ7 6から取得されたデータは、1 6ビット幅の入力F I F Oを介してA L U 1 8 8へ転送される。出力データは、1 6ビット幅の出力F I F Oに書き込まれ、そこからデータキャッシュ7 6へ書き込まれる。最後に、全ての処理ユニット、例えば、P U 1 7 8は、単一の8ビット幅V L I W入力F I F O 7 8と、単一の8ビット幅V L I W出力F I F O 7 9を共用する。低速データバス接続によって、C P Uは、処理ユニット、例えば、P U 1 7 8のレジスタを読み書きし、マイクロコードを更新し、V L I Wベクトルプロセッサ7 4の全ての処理ユニット、例えば、P U 1 7 8によって共用される共通レジスタを読み書きすることが可能になる。次に、図4を参照すると、単独の処理ユニット、例えば、P U 1 7 8の内部のより詳細が示され、以下では、コンポーネント及び制御信号を詳述する。

20

30

40

【0 2 3 5】

マイクロコード

各処理ユニット、例えば、P U 1 7 8は、その特定の処理ユニット、例えば、P U 1 7 8用のプログラムを保持するマイクロコードR A M 1 9 6を含む。マイクロコードをR O Mに収容するのではなく、マイクロコードはR A Mに収容され、C P Uはマイクロコードを

50

アップロードする役割を担う。チップ上の同じ空間であるため、このトレードオフは、ある一つの機能の最大サイズをRAM 196のサイズまで縮小するが、マイクロコードに書き込まれる機能の数は無制限である。マイクロコードを用いて実現された機能は、Varkアクセラレーション、Artcard読み取り、及び印刷を含む。VLIWベクトルプロセッサ74の方式は、ACP31のケースでは、以下の幾つかの利点がある：

ハードウェア設計の複雑さが低減される；

ハードウェアのリスクは複雑さの低減によって低減する；

ハードウェア設計時間は、専用シリコンに組み込まれる全てのVark機能に依存しない；

チップ上の空間が全体的に縮小される（多数のプロセスをマイクロコードとして組み込むことができるため）；

ハードウェア設計時間に影響を与えることなく、（マイクロコードによって）Varkに機能を加えることができる。

【0236】

サイズ及び内容

各処理ユニット、例えば、PU178を制御するCPUによってロードされたマイクロコードRAM 196は128ワードであり、各ワードは96ビット幅である。処理ユニット、例えば、PU178の様々なユニットを制御するためのマイクロコードサイズの概要は以下の表に掲載されている。

【0237】

【表5】

プロセスブロック	サイズ(ビット数)
状態出力	3
分岐(マイクロコード制御)	11
入 力	8
出 力	6
レジスタ	7
読み出し	10
書き込み	6
バレルシフタ	12
アダー／ロジック	14
乗算／補間	19
合 計	96

128個の命令ワードによって、処理ユニット、例えば、PU178毎の全マイクロコードRAM 196は、12288ビット、即ち、正確に1.5KBである。VLIWベクトルプロセッサ74は、4個の同一の処理ユニット、例えば、PU178により構成されるので、これは、6144バイト、即ち、正確に6KBと一致する。マイクロコードワード内の一部のビットは、制御ビットとしてそのまま使用され、一方、他のビットはデコードされる。マイクロコードワードのビットの各々の解釈を詳細に記述する種々のユニット説明を参照せよ。

【0238】

処理ユニット、例えば、PU178の間の同期

各処理ユニット、例えば、PU178は、4ビット同期レジスタ197を含む。それは、一体的に動作する処理ユニット、例えば、PU178を決定するため使用されるマスクであり、単一プロセスとして機能する対応した処理ユニット、例えば、PU178の各々に対して1ビットがセットされる。例えば、全ての処理ユニット、例えば、PU178が単一プロセスとして機能する場合、4個の同期レジスタ197の各々は、全て4ビットがセ

ットされるであろう。2 個の処理ユニット、例えば、PU 178 毎に二つの同期プロセスが存在するならば、2 個の処理ユニット、例えば、PU 178 は、（その PU 自体に対応した）同期レジスタ内の 2 ビットがセットされ、他の 2 個の処理ユニットは、（その PU 自体に対応した）同期レジスタ内の別の 2 ビットがセットされる；

同期レジスタ 197 は、以下の二つの基本的な方式で使用される；

同期した所定のプロセスを停止及び開始する；

プロセス内で実行を一時停止する。

【0239】

停止及び開始プロセス

CPU は、マイクロコード RAM 196 にロードし、第 1 の命令の実行アドレス（通常は 0）をロードする役割を担う。CPU がマイクロコードを実行し始めるとき、マイクロコードは指定されたアドレスから始まる。

【0240】

マイクロコードが実行されるのは、同期レジスタ 197 の全てのビットが共通同期レジスタ 197 にもセットされたときに限られる。したがって、CPU は、全ての処理ユニット、例えば、PU 178 をセットアップし、次に、共通同期レジスタ 197 への 1 回の書き込みでプロセスを開始又は停止する。

【0241】

この同期方式は、複数のプロセスが処理ユニット、例えば、PU 178 上で非同期的に実行されることを可能にさせ、同時に 1 個の処理ユニット、例えば、PU 178 としてではなく、プロセスとして停止及び開始される、

プロセス内の実行の一時停止

所与のサイクルにおいて、処理ユニット、例えば、PU 178 は、（現在のマイクロコード命令のオペレーションコードに基づいて）FIFO からの読み出し、又は FIFO への書き込みのために必要である。FIFO が読み出し要求時にエンプティであるか、又は書き込み要求時にフルである場合、FIFO 要求は完了し得ない。処理ユニット、例えば、PU 178 は、したがって、プロセス一時停止制御信号 198 をアサートする。全ての処理ユニット、例えば、PU 178 からのプロセス一時停止信号は、全ての処理ユニット、例えば、PU 178 へフィードバックされる。同期レジスタ 197 は、4 個のプロセス一時停止ビットと論理積演算され、結果が非零であるならば、処理ユニット、例えば、PU 178 のレジスタのライトイネーブル、又は FIFO ストロープはセットされない。したがって、そのタスクを終了できなかった処理ユニット、例えば、PU 178 と同じプロセスグループを形成する処理ユニット、例えば、PU 178 の中に、自分のレジスタ又は FIFO がそのサイクル中に更新された処理ユニットは存在しない。この簡単な技術は、所定のプロセスグループを同期状態に保つ。後の各サイクルで、処理ユニット、例えば、PU 178 の状態機械は、同じアドレスのこのマイクロコード命令を再実行することを試み、成功するまでそのようにし続ける。勿論、共通同期レジスタ 197 は、必要に応じてプロセス全体を停止させるため、CPU によって書き込むことができる。この同期方式は、任意の処理ユニット、例えば、PU 178 の組み合わせが一体となって動作することを可能にさせ、各グループは、データを読み書きする準備ができていないことを原因とする一時停止に関しては、協働して動作する処理ユニットだけに影響を与える。

【0242】

制御及び分岐

各サイクル中、処理ユニット、例えば、PU 178 の ALU 188 内の 4 個の基本入力及び計算ユニットの各々（読み出し、アダー／ロジック、乗算／補間、及びバレルシフタ）は、そのサイクル中の演算結果が 0 であるか又は負であるかを示す零フラグ及び負フラグの二つの状態ビットを生成する。各サイクルに、これらの 4 個の状態ビットのうちの一つが、処理ユニット、例えば、PU 178 から出力されるマイクロコード命令によって選択される。4 個の状態ビット（一つの処理ユニット、例えば、PU 178 の ALU 188 について 1 個）が 4 ビット共通状態レジスタ 200 に組み合わせられる。次のサイクル中に、

各処理ユニット、例えば、PU178のマイクロコードプログラムは、共通状態レジスタ200からのビットのうちの1ビットを選択し、状態ビットの値に依存して別のマイクロコードアドレスへ分岐することが可能である。

【0243】

状態ビット

各処理ユニット、例えば、PU178のALU188は、多数の入力及び計算ユニットを含む。各ユニットは、負フラグと零フラグの二つの状態ビットを生成する。これらの状態ビットのうちの一つは、特別なユニットが1ビット3状態の状態ビットバス上の値をアサートするときに、処理ユニット、例えば、PU178から出力される。一つの状態ビットが処理ユニット、例えば、PU178から出力され、次に、共通状態レジスタ200を更新するため、他の処理ユニット、例えば、PU178の状態ビットと組み合わせられる。出力状態ビットを決定するマイクロコードは以下の形式をとる。

【0244】

【表6】

ビット数	説 明
2	出力される状態ビットを有するユニットを選択 00 = アダーユニット 01 = 乗算／論理ユニット 10 = バレルシフトユニット 11 = 読み出しユニット
1	0 = 零フラグ 1 = 負フラグ
3	合 計

ALU188内で、2ビット選択プロセッサブロック値は、4個の1ビットイネーブルビットにデコードされ、異なるイネーブルビットが各プロセッサユニットブロックへ送信される。状態選択ビット（零又は負を選択する）は、状態ビットバスへ出力されるべきビットを決定するため全てのユニットへ渡される。

【0245】

マイクロコード内の分岐

各処理ユニット、例えば、PU178は、実行中の現在のマイクロコードアドレスを保持する7ビットプログラムカウンタ（PC）を含む。普通のプログラム実行はリニアであり、即ち、あるサイクルのアドレスNから次のサイクルのアドレスN+1へ進む。しかし、サイクル毎に、マイクロコードプログラムは、異なる場所へ分岐する能力、又は共通状態レジスタ200からの状態ビットをテストして分岐する能力を備えている。次の実行アドレスを決定するマイクロコードは以下の形式をとる。

【0246】

10

20

30

【表 7】

ビット数	説 明
2	00 = ノップ (PC = PC + 1) 01 = 常に分岐 10 = 状態ビットがクリアのとき分岐 11 = 状態ビットがセットのとき分岐
2	状態ワードから状態ビットを選択
7	分岐先アドレス(絶対アドレス、00 - 7F)
11	合 計

10

A L U 1 8 8

図 5 は A L U 1 8 8 を詳細に示す図である。A L U 1 8 8 内部には、多数の専用処理ブロックが存在し、マイクロコードプログラムによって制御される。専用処理ブロックには、入力 F I F O からのデータを受け取る読み出しブロック 2 0 2 と、出力 F I F O を介してデータを送り出す書き込みブロック 2 0 3 と、加算及び減算と、比較及び論理演算のためのアダー／ロジックブロック 2 0 4 と、乗算タイプの補間及び乗算／累算のための乗算／補間ブロック 2 0 5 と、要求に応じてデータをシフトするバレルシフトブロック 2 0 6 と、外部クロスバースイッチ 1 8 3 からのデータを受け取る入力ブロック 2 0 7 と、外部クロスバースイッチ 1 8 3 へデータを送り出す出力ブロック 2 0 8 と、一次記憶装置にデータを保持するレジスタブロック 2 1 5 と、が含まれる。

20

【 0 2 4 7 】

4 個の専用 3 2 ビットレジスタは、4 個の主要な処理ブロックの結果を保持する：
M レジスタ 2 0 9 は、乗算／補間ブロックの結果を保持する；
L レジスタ 2 0 9 は、アダー／ロジックブロックの結果を保持する；
S レジスタ 2 0 9 は、バレルシフトブロックの結果を保持する；
R レジスタ 2 0 9 は、読み出しブロック 2 0 2 の結果を保持する。

30

【 0 2 4 8 】

更に、データ転送用の二つの内部クロスバースイッチ 2 1 3 及び 2 1 4 が存在する。多様な処理ブロックは、各ブロックに属するマイクロコード定義と共に後続のセクションで更に展開される。尚、マイクロコードは、内部の様々なユニットへ制御信号を供給するためブロック内でデコードされる。

【 0 2 4 9 】

処理ユニット、例えば、P U 1 7 8 間のデータ転送

各処理ユニット、例えば、P U 1 7 8 は、外部クロスバーを介してデータを交換することができる。処理ユニット、例えば、P U 1 7 8 は、二つの入力を取得し、2 個の値を外部クロスバーへ出力する。このようにして、処理用の 2 個のオペランドが単一サイクルで獲得できるが、実際には、次のサイクルまで演算に使用されない。

40

【 0 2 5 0 】

入力 2 0 7

このブロックは図 6 に示され、外部クロスバーからのデータを受け取る 2 個のレジスタ I n 1 及び I n 2 を含む。レジスタはサイクル毎にロードすることが可能であり、又はそのままの状態を保つことが可能である。8 入力から選択するための選択ビットは、外部クロスバースイッチ 1 8 3 へ出力される。マイクロコードは以下の形式をとる。

【 0 2 5 1 】

【表 8】

ビット数	説 明
1	0 = ノップ 1 = クロスバーから In ₁ へロード
3	外部クロスバーから入力 1 を選択
1	0 = ノップ 1 = 外部クロスバーから In ₂ へロード
3	外部クロスバーから入力 2 を選択
8	合 計

10

出力 2 0 8

入力 2 0 8 によって補完される。出力ブロックは図 7 に詳細に示されている。出力は、Out₁ 及び Out₂ の 2 個のレジスタを含み、両方のレジスタは、他の処理ユニット、例えば、PU 1 7 8 によって使用するためサイクル毎に外部クロスバーへ出力される。また、書き込みユニットは、Out₁ と Out₂ のうちの一方を、ALU 1 8 8 に取り付けられた出力 FIFO の一つに書き込むことが可能である。最終的に、両方のレジスタは、第 1 クロスバー 2 1 3 への入力として利用可能であり、第 1 クロスバー 2 1 3 は、そのレジスタ値が ALU 1 8 8 内の他のユニットへの入力として利用できるようにする。各サイクルで、2 個のレジスタのうち何れか一方は、マイクロコード選択に応じて更新することができる。指定されたレジスタにロードされたデータは、（第 1 クロスバー 2 1 3 から選択された）D₀ - D₃ のうちの一つ、（第 2 クロスバー 2 1 4 から選択された）M、L、S 及び R のうちの一つ、2 個のプログラマブル定数のうちの一つ、又は固定値 0 若しくは 1 である。出力用のマイクロコードは以下の形式をとる。

20

【0 2 5 2】

【表 9】

ビット数	説 明
1	0 = ノップ 1 = レジスタにロード
1	ロード先のレジスタを選択 [Out ₁ 又は Out ₂]
4	入力を選択 [In ₁ , In ₂ , Out ₁ , Out ₂ , D ₀ , D ₁ , D ₂ , D ₃ , M, L, S, R, K ₁ , K ₂ , 0, 1]
6	合 計

30

ALU 1 8 8 内のローカルレジスタ及びデータ転送

前述の通り、ALU 1 8 8 は、以下の 4 個のブロックの結果を保持するため、4 個の専用 3 2 ビットレジスタを含む：

M レジスタ 2 0 9 は乗算／補間ブロックの結果を保持する；

L レジスタ 2 0 9 はアダー／ロジックブロックの結果を保持する；

S レジスタ 2 0 9 はバレルシフタブロックの結果を保持する；

R レジスタ 2 0 9 は読み出しブロック 2 0 2 の結果を保持する。

40

【0 2 5 3】

CPU はこれらのレジスタに直接アクセスを行い、他のユニットは、第 2 クロスバー 2 1 4 経由の入力としてそれらを選択することができる。場合によっては、演算を 1 サイクル以上遅延させる必要がある。レジスタブロックは、4 個の 3 2 ビットレジスタ D₀ - D₃ を含み、処理中の一時変数を保持する。各サイクルに、レジスタのうちの一つを更新可能であり、一方、全てのレジスタは、他のユニットが（同様に、In₁、In₂、Out₁ 及び Out₂ を含む）第 1 クロスバー 2 1 3 を介して使用するために出力される。CPU はこれらのレジスタへ直接アクセスを行う。指定されたレジスタにロードされたデータは、（

50

第 1 クロスバー 2 1 3 から選択された) $D_0 - D_3$ のうちのの一つ、(第 2 クロスバー 2 1 4 から選択された) M、L、S 及び R のうちのの一つ、2 個のプログラマブル定数のうちのの一つ、又は固定値 0 若しくは 1 である。レジスタブロック 2 1 5 は図 8 に詳細に示されている。レジスタ用のマイクロコードは以下の形式をとる。

【0 2 5 4】

【表 1 0】

ビット数	説 明
1	0 = ノップ 1 = レジスタにロード
2	ロード先のレジスタを選択 [$D_0 - D_3$]
4	入力を選択 [$In_1, In_2, Out_1, Out_2, D_0, D_1, D_2, D_3, M, L, S, R, K_1, K_2, 0, 1$]
7	合 計

10

第 1 クロスバー 2 1 3

第 1 クロスバー 2 1 3 は図 9 に詳細に示されている。第 1 クロスバー 2 1 3 は、入力 In_1 、 In_2 、 Out_1 、 Out_2 、 $D_0 - D_3$ から選択するため使用される。7 個の出力が第 1 クロスバー 2 1 3 から発生され、3 個が乗算／補間ユニットへ、2 個がアダーユニットへ、1 個がレジスタユニットへ、1 個が出力ユニットへ向けられる。第 1 クロスバー 2 1 3 用の制御信号は、クロスバー入力を使用する様々なユニットから到来する。第 1 クロスバー 2 1 3 のため分離された特定のマイクロコードは存在しない。

20

【0 2 5 5】

第 2 クロスバー 2 1 4

第 2 クロスバー 2 1 4 は図 1 0 に詳細に示されている。第 2 クロスバー 2 1 4 は、汎用 ALU 1 8 8 レジスタ M、L、S 及び R から選択するため使用される。6 個の出力が第 1 クロスバー 2 1 3 から発生され、2 個が乗算／補間ユニットへ、2 個がアダーユニットへ、1 個がレジスタユニットへ、1 個が出力ユニットへ向けられる。第 2 クロスバー 2 1 4 用の制御信号は、クロスバー入力を使用する様々なユニットから到来する。第 2 クロスバー 2 1 4 のため分離された特定のマイクロコードは存在しない。

【0 2 5 6】

処理ユニット、例えば、PU 1 7 8 と DRAM 又は外部プロセスとの間のデータ転送
図 4 を参照すると、処理ユニット、例えば、PU 1 7 8 は、外部クロスバーを介して、直接的に相互にデータを共有する。また、処理ユニットは、外部プロセス並びに DRAM との間でデータを転送する。各処理ユニット、例えば、PU 1 7 8 は、2 個の I/O アドレス生成器 1 8 9 及び 1 9 0 を含み、DRAM との間でデータを転送する。処理ユニット、例えば、PU 1 7 8 は、I/O アドレス生成器の出力 FIFO、例えば、1 8 6 を介して DRAM へデータを送出し、又は I/O アドレス生成器の入力 FIFO 1 8 7 を介して、DRAM からデータを受け取る。これらの FIFO は、処理ユニット、例えば、PU 1 7 8 に局在する。また、外部プロセスとの間で、全ての ALU によって共有される共通 VLIW 入力 FIFO 7 8 及び共通 VLIW 出力 FIFO 7 9 の形でデータを送受する機構が存在する。VLIW 入力及び出力 FIFO は、8 ビット幅しかなく、印刷、Art card 読み出し、CPU 等へのデータ転送に使用される。ローカル入力及び出力 FIFO は 1 6 ビット幅である。

30

40

【0 2 5 7】

読み出し

図 5 の読み出しプロセスブロック 2 0 2 は、ALU 1 8 8 の R レジスタ 2 0 9 を更新する役割を担い、R レジスタは、VLIW マイクロコード化プロセスへの外部入力データを表現する。サイクル毎に、読み出しユニットは、共通 VLIW 入力 FIFO 7 8 (8 ビット)、又は 2 個のローカル入力 FIFO (1 6 ビット) のうちの何れかから読み出すことが可能である。3 2 ビット値が生成され、そのデータの全部又は一部が R レジスタ 2 0 9 へ

50

転送される。プロセスは図 1 1 に示されている。読み出し用のマイクロコードは以下の表に掲載されている。一部のビットパターンの解釈はデコーディングを補助するため意図的に選択される。

【 0 2 5 8 】

【表 1 1】

ビット数	説 明
2	00 = ノップ 01 = VLIW入力 FIFO78 から読み出し 10 = ローカル FIFO1 から読み出し 11 = ローカル FIFO2 から読み出し
1	有意ビット数 0 = 8 ビット (0 でパッド又は符号拡張) 1 = 16 ビット (ローカル FIFO 読み出しの場合に限り有効)
1	0 = 符号無しとしてデータ取扱 (0 でパッド) 1 = 符号付きとしてデータ取扱 (FIFO からの読み出し時に符号拡張)
2	データ左シフト量 00 = 0 ビット (変化なし) 01 = 8 ビット 10 = 16 ビット 11 = 24 ビット
4	R の更新対象のバイト (高次から低次のバイト) 4 ビットの各々は R 上の 1 バイトの書き込みイネーブル(WriteEnable)を表現
10	合 計

書き込み

書き込みプロセスブロックは、各サイクルに、共通 V L I W出力 F I F O 7 9、又は 2 個のローカル出力 F I F O の一方の何れかへの書き込みを行うことができる。所与のサイクル中に書き込まれる F I F O は 1 個しかないので、1 個の 1 6 ビット値が全ての F I F O へ出力され、下位 8 ビットが V L I W出力 F I F O 7 9 へ達することに注意する必要がある。マイクロコードは、どの F I F O がその値を受け付けるかを制御する。データ選択のプロセスは図 1 2 により詳細に示されている。ソース値 O u t₁ 及び O u t₂ は出力ブロックから到来する。それらは、単に 2 個のレジスタである。書き込み用のマイクロコードは以下の形式をとる。

【 0 2 5 9 】

10

20

30

【表 1 2】

ビット数	説 明
2	00 = ノップ 01 = VLIW出力 FIFO79 へ書き込み 10 = ローカル出力 FIFO1 へ書き込み 11 = ローカル出力 FIFO2 へ書き込み
1	出力値選択 [Out ₁ 又は Out ₂]
3	書き込む出力値の部分を選択(32 ビット = 4 バイト ABCD) 000 = 0D 001 = 0D 010 = 0B 011 = 0A 100 = CD 101 = BC 110 = AB 111 = 0
6	合 計

10

計算ブロック

各 A L U 1 8 8 は、2 個の計算プロセスブロック、即ち、アダー／ロジックプロセスブロック 2 0 4 と、乗算／補間プロセスブロック 2 0 5 と、を含む。更に、これらの計算ブロックを補助するためバレルシフタブロックが設けられる。レジスタブロック 2 1 5 からのレジスタは、パイプライン演算中に一時記憶装置として使用することも可能である。

【0 2 6 0】

バレルシフタ

バレルシフタプロセスブロック 2 0 6 は図 1 3 に詳細に示され、その入力は、アダー／ロジック、若しくは、乗算／補間プロセスブロックの出力、又はそれらのブロック（A L U レジスタ L 及び M）の前のサイクルの結果から得られる。選択された 3 2 ビットは、（必要に応じて符号を拡張して）何れかの向きに任意のビット数だけバレルシフトされ、A L U 1 8 8 の S レジスタ 2 0 9 へ出力される。バレルシフトプロセスブロック用のマイクロコードは以下の表に記載されている。一部のビットパターンの解釈はデコーディングを助けるため意図的に選択されることに注意する必要がある。

【0 2 6 1】

20

30

【表 1 3】

ビット数	説 明
3	000 = ノップ 001 = 左シフト (符号無し) 010 = 予備 011 = 左シフト (符号付き) 100 = 右シフト (符号無し、丸め無し) 101 = 右シフト (符号無し、丸め付き) 110 = 右シフト (符号付き、丸め無し) 111 = 右シフト (符号付き、丸め付き)
2	バレルシフトへの入力選択 00 = 乗算／補間結果 01 = M 10 = アダー／ロジック結果 11 = L
5	シフトビット数
1	255 の上限
1	0 の下限 (符号付きデータ)
12	合 計

10

20

30

アダー／ロジック 2 0 4

アダー／ロジックプロセスブロックは、図 1 4 により詳細に示され、簡単な 3 2 ビット加算／減算、比較、及び論理演算用に設計されている。単一サイクルで、1 回の加算、比較、又は論理演算を実行することができ、その結果は A L U 1 8 8 の L レジスタ 2 0 9 に保存される。二つの主オペランド A 及び B が存在し、これらは、二つのクロスバーの何れか、又は 4 個の定数レジスタから選択される。一方のクロスバーの選択によって、前のサイクルの算術演算の結果を使用することが可能になり、もう一方は、この A L U 又は他の A L U 1 8 8 によって前に計算されたオペランドへアクセスする。C P U は、4 個の定数 ($K_1 - K_4$) にアクセスする唯一のユニットである。例えば、 $(A + B) \times 4$ のような演算が要求される場合、アダーからの直接出力は、バレルシフタへの入力として使用可能であり、先に L レジスタ 2 0 9 にラッチしなくても 2 個分だけ左シフトさせることができる。アダーからの出力は、乗算－累算演算のため乗算ユニットで利用することができる。アダー／ロジックプロセスブロック用のマイクロコードは以下の表に掲載されている。一部のビットパターンの解釈はデコーディングを補助するため意図的に選択される。

アダー／ロジックユニット用のマイクロコードビット解釈

【 0 2 6 2 】

【表 1 4】

ビット数	説 明
4	0000 = $A + B$ (キャリー入力 = 0) 0001 = $A + B$ (キャリー入力 = 前の演算のキャリー出力) 0010 = $A + B + 1$ (キャリー入力 = 1) 0011 = $A + 1$ (A をインクリメント) 0100 = $A - B - 1$ (キャリー入力 = 0) 0101 = $A - B$ (キャリー入力 = 前の演算のキャリー出力) 0110 = $A - B$ (キャリー入力 = 1) 0111 = $A - 1$ (A をデクリメント) 1000 = ノップ 1001 = $ABS(A - B)$ 1010 = $MIN(A, B)$ 1011 = $MAX(A, B)$ 1100 = $A AND B$ (A 及び B は共に反転可能、以下を参照) 1101 = $A OR B$ (A 及び B は共に反転可能、以下を参照) 1110 = $A XOR B$ (A 及び B は共に反転可能、以下を参照) 1111 = A (A は反転可能、以下を参照)
1	論理演算の場合 : 0 = $A = A$ 1 = $A = NOT(A)$ アダー演算の場合 : 0 = A は符号無し 1 = A は符号付き
1	論理演算の場合 : 0 = $B = B$ 1 = $B = NOT(B)$ アダー演算の場合 : 0 = B は符号無し 1 = B は符号付き
4	A を選択 [In ₁ , In ₂ , Out ₁ , Out ₂ , D ₀ , D ₁ , D ₂ , D ₃ , M, L, S, R, K ₁ , K ₂ , K ₃ , K ₄]
4	B を選択 [In ₁ , In ₂ , Out ₁ , Out ₂ , D ₀ , D ₁ , D ₂ , D ₃ , M, L, S, R, K ₁ , K ₂ , K ₃ , K ₄]
14	合 計

10

20

30

40

50

乗算／補間 2 0 5

乗算／補間プロセスブロックは、図 1 5 に詳細に示され、4 個の 8×8 形補間ユニットのセットであり、4 個の 8×8 形補間ユニットは、1 サイクル毎に 4 回の別個の 8×8 形補間を実行する能力を備えているか、又は 1 回の 16×16 形乗算を実行するため組み合わせることができる。これは、単一サイクル中に最大で 4 回のリニア補間、1 回のバイリニア補間、又はトライリニア補間の半分を実行可能であることを示す。補間又は乗算の結果は A L U 1 8 8 の M レジスタ 2 0 9 に保存される。二つの主オペランド A 及び B は、A L U 1 8 8 の汎用レジスタの何れかから選択されるか、又は乗算／補間プロセスブロックの

内部にある 4 個のプログラマブル定数から選択される。各補間ブロックは、簡単な 8 ビット補間器 [結果 = $A + (B - A) f$] として機能するか、又は簡単な 8×8 形乗算 [結果 = $A * B$] として機能する。演算が補間である場合、A 及び B は、4 個の 8 ビット数値 A_0 から A_3 (A_0 は下位バイトである)、及び B_0 から B_3 として取り扱われる。A G E N、B G E N 及び F G E N は、実行される演算に適合するように補間ユニットへの入力を並べる役目を果たす。例えば、バイリニア補間を実行するため、4 個の値の各々は、異なる係数によって乗算され、その結果が加算され、一方、 16×16 形の乗算は、係数を 0 にする必要がある。アダー／ロジックプロセスブロック用のマイクロコードは、以下の表に掲載されている。尚、一部のビットパターンの解釈はデコーディングを補助するため意図的に選択される。

【 0 2 6 3 】

【表 1 5】

ビット数	説 明
4	$0000 = (A_{10} * B_{10}) + V$ $0001 = (A_0 * B_0) + (A_1 * B_1) - V$ $0010 = (A_{10} * B_{10}) - V$ $0011 = V - (A_{10} * B_{10})$ $0100 = f_0$ による A_0, B_0 の補間 $0101 = f_0$ による A_0, B_0 と f_1 による A_1, B_1 の補間 $0110 = f_0$ による A_0, B_0 と f_1 による A_1, B_1 と f_2 による A_2, B_2 の補間 $0111 = f_0$ による A_0, B_0 と f_1 による A_1, B_1 と f_2 による A_2, B_2 と f_3 による A_3, B_3 の補間 $1000 = 16$ ビット補間のステージ 1 [$M = A_{10} * f_{10}$] $1001 = 16$ ビット補間のステージ 2 [$M = M + (A_{10} * f_{10})$] $1010 = f$ による A のトライリニア補間のステージ 1 $[M = A_0 f_0 + A_1 f_1 + A_2 f_2 + A_3 f_3]$ $1011 = f$ による A のトライリニア補間のステージ 2 $[M = M + A_0 f_0 + A_1 f_1 + A_2 f_2 + A_3 f_3]$ $1100 = f$ による A のバイリニア補間のステージ 1 [$M = A_0 f_0 + A_1 f_1$] $1101 = f$ による A のバイリニア補間のステージ 2 $[M = M + A_0 f_0 + A_1 f_1]$ $1100 = f$ による A のバイリニア補間の完成 $[M = A_0 f_0 + A_1 f_1 + A_2 f_2 + A_3 f_3]$ $1111 = \text{ノップ}$
4	A を選択 [$In_1, In_2, Out_1, Out_2, D_0, D_1, D_2, D_3, M, L, S, R, K_1, K_2, K_3, K_4$]
4	B を選択 [$In_1, In_2, Out_1, Out_2, D_0, D_1, D_2, D_3, M, L, S, R, K_1, K_2, K_3, K_4$]
乗算の場合 :	
4	V を選択 [$In_1, In_2, Out_1, Out_2, D_0, D_1, D_2, D_3, K_1, K_2, K_3, K_4, \text{アダー結果}, M, 0, 1$]
1	符号付きとして A を取扱
1	符号付きとして B を取扱
1	符号付きとして V を取扱
補間の場合 :	
4	f の基底を選択 [$In_1, In_2, Out_1, Out_2, D_0, D_1, D_2, D_3, K_1, K_2, K_3, K_4, X, X, X, X$]
1	P_1 又は P_2 から補間 f 生成を選択 P_n は f の小数ビット数として解釈される $P_n = 0$ の場合、f は 0...1 を表現するレンジ 0..255 である
2	予 備
19	合 計

同じ 4 ビットが V 及び f の選択のために使用されるが、V の直前の 4 個の選択肢は、一般的に f 値として意味をなさない。1 又は 0 の係数を用いる補間は無意味であり、前の乗算若しくは現在の結果が f に対する有意な値である可能性は少ない。

【 0 2 6 4 】

I / O アドレス生成器 1 8 9 と 1 9 0

10

20

30

40

50

I/Oアドレス生成器は図16に詳細に示されている。VLIWプロセスはDRAMに直接アクセスしない。アクセスは2個のI/Oアドレス生成器189、190を介して行われ、各々のI/Oアドレス生成器は固有の入力FIFO及び出力FIFOを含む。処理ユニット、例えば、PU178は、2個のローカル入力FIFOの一方からデータを読み出し、2個のローカル出力FIFOの一方にデータを書き込む。各I/Oアドレス生成器は、DRAMからデータを読み出し、そのデータを入力FIFOに入れる責任があり、その入力FIFOにおいて、データは処理ユニット、例えば、PU178によって読み出すことができる。各I/Oアドレス生成器は、その出力FIFOから（処理ユニット、たとえば、PU178によって入れられた）データを取り出し、それをDRAMに書き込む役割を担う。I/Oアドレス生成器は、データキャッシュ76を介してDRAMのデータを取

10

得及び記憶するためのアドレスを生成し制御を行う状態機械である。それは、CPUソフトウェア制御の下でカスタマイズ可能であるが、マイクロコード化できない。アドレス生成器は、

多種多様な方式で画像の画素の最初から最後までを繰り返すため（読み、書き、もしくはその両方）使用される画像イタレータと、

画像の画素、テーブルのデータをランダムにアクセスし、DRAMのFIFOをシミュレートするため使用されるテーブルI/Oと、

の二つに大別されるアドレスを生成する。

【0265】

I/Oアドレス生成器189と190の各々は、データキャッシュ76への固有のバス配線を具備し、1個の処理ユニット、例えば、PU178について2本のバス配線があり、VLIWベクトルプロセッサ74全体で合計8本のバスがある。データキャッシュ76は、サイクル毎に、4個の処理ユニット、例えば、PU178からの最大で8個の要求のうちの4個のために作用することが可能である。入力FIFO及び出力FIFOは、深さが8エントリーで、幅が16ビットのFIFOである。多種多様のアドレス生成（画像イタレータ及びテーブルI/O）が後のセクションで説明される。

20

【0266】

レジスタ

I/Oアドレス生成器は、アドレス生成を制御するため使用されるレジスタの組を有する。また、アドレッシングモードは、データがどのようにフォーマットされ、ローカル入力FIFOへ送られるか、並びに、データが出力FIFOからどのように解釈されるかを決定する。CPUは、低速バスを介して、I/Oアドレス生成器のレジスタにアクセス可能である。第1のレジスタの組は、I/O生成器のためのハウスキーピングパラメータを定義する。

30

【0267】

【表 1 6】

レジスタ名	ビット数	説 明
リセット	0	このレジスタへの書き込みによってあらゆる動作が停止し、I/O 生成器の全てのデータレジスタに 0 が書き込まれる。入力及び出力 FIFO はクリアされない。
実 行	0	このレジスタへの書き込みによって現在のセットアップによるカウンタが再起動する。例えば、I/O 生成器が読み出しイタレータであり、イタレータが現在画像の途中にある場合、実行レジスタへの書き込みによって、読み出しが画像の先頭から再開する。I/O 生成器が実行中である間、状態レジスタの動作中ビットがセットされる。
停 止	0	このレジスタへの書き込みによって現在の動作を停止し、状態レジスタの動作中ビットがクリアされる。動作中ビットが既にクリアされている場合、このレジスタへの書き込みによる効果はない。
再 開	0	このレジスタへの書き込みは、現在のセットアップから I/O 生成器を再開させる。カウンタはリセットされず、FIFO はクリアされない。I/O 生成器が動作中である間に、このレジスタへの書き込みによる効果はない。
実行ビット FIFO クリア	1	0 = 実行ビットへの書き込み時に FIFO をクリアしない 1 = 実行ビットへの書き込み時に FIFO をクリアする
状 態	8	状態フラグ

10

20

状態レジスタは以下の値をとる。

【 0 2 6 8】

【表 1 7】

レジスタ名	ビット数	説 明
動作中	1	0 = 現在非動作中 1 = 現在動作中
予 備	7	—

30

キャッシュ処理

複数のレジスタがキャッシュ処理機構を制御するため使用され、入力、出力等に使用するキャッシュグループを指定する。キャッシュグループのより詳細な情報についてはデータキャッシュ 7 6 に関するセクションを参照のこと。

【 0 2 6 9】

【表 1 8】

レジスタ名	ビット数	説 明
キャッシュグループ 1	4	データ読み出し元のキャッシュグループを定義
キャッシュグループ 2	4	データ書き込み先のキャッシュグループを定義し、画像ピラミッドルックアップ I/O モードの場合、レベル情報テーブルを読み出すために使用するキャッシュを定義。

40

画像イタレータ＝順次自動画素アクセス

ソフトウェア及びハードウェアアルゴリズム用の主な画像絵素アクセス方法は画像イタレ

50

ータによる。画像イタレータは、画像チャンネル内の画素のキャッシュのアドレッシング及びアクセスの全てを実行し、それらのクライアントのため画素を読み出し、書き込み、或いは、読み書きする。読み出しイタレータは、そのクライアントのための特定の順序で画素を読み出し、書き込みイタレータはそのクライアントのための特定の順序で画素を書き込む。イタレータのクライアントは、ローカル入力 F I F O から画素を読み出し、又はローカル出力 F I F O を介して画素を書き込む。

【0270】

読み出し画像イタレータは、特定の順序で画像の中を読み、画素データをローカル入力 F I F O に入れる。クライアントが入力 F I F O から画素を読み出す毎に、読み出しイタレータは（データキャッシュ76経由で）画像からの次の画素を F I F O に入れる。

10

【0271】

書き込み画像イタレータは、画像全体を書き出すため、特定の順序で画素を書き込む。クライアントは画素を出力バッファに書き込み、この出力バッファが次に書き込み画像イタレータによって読み出され、データキャッシュ76を介して D R A M に書き込まれる。

【0272】

典型的に、V L I W プロセッサは、読み出しイタレータに結合された入力と、対応した書き込みイタレータに結合された出力と、を有する。処理ユニット、例えば、P U 1 7 8 のマイクロプログラムの観点では、F I F O は、D R A M への効率的なインタフェースである。記憶を（データの論理的な順序ではなく）実際に実行する方法は無関係である。F I F O は、長さが実質的に制限されていると考えられるが、實際上、F I F O は長さが有限であり、特に、複数のメモリアクセスが競合する場合、データをソートし、取り出す遅延が生じる。画像処理アルゴリズムの最も一般的なアドレッシング必要条件に対処する様々な画像イタレータが存在する。殆どの場合に、各読み出しイタレータには、対応した書き込みイタレータが存在する。種々のイタレータが以下の表に掲載されている。

20

【0273】

【表19】

読み出しイタレータ	書き込みイタレータ
順次読み出し	順次書き込み
ボックス読み出し	—
垂直ストリップ読み出し	垂直ストリップ書き込み

30

4 ビットアドレスモードレジスタがイタレータタイプを決定するため使用される。

【0274】

【表20】

ビット数	アドレスモード
3	0 = このアドレッシングモードはイタレータである
2 から 0	イタレータモード 001 = 順次イタレータ 010 = ボックス [読み出し専用] 100 = 垂直ストリップ 残りのビットパターンは予備

40

アクセス専用レジスタは以下のように使用される。

【0275】

【表 2 1】

レジスタ名	ローカル名	説 明
アクセス専用 ₁	フラグ	読み出し及び書き込みのため使用されるフラグ。
アクセス専用 ₂	X ボックスサイズ	ボックス読み出しの X のサイズを決定する。有効値は 3, 5 及び 7 である。
アクセス専用 ₃	Y ボックスサイズ	ボックス読み出しの Y のサイズを決定する。有効値は 3, 5 及び 7 である。
アクセス専用 ₄	ボックスオフセット	ボックス読みだし専用中のある画素中心と次との間のオフセット。 通常値は 1 であるが、その他の有効値には、2, 4, 8...が含まれる。 詳細はボックス読み出しを参照のこと。

10

フラグレジスタ（アクセス専用 1）は、データの読み出し及び書き込みに影響を与える要因を決定する多数のフラグを含む。フラグレジスタは以下のように構成される。

【0 2 7 6】

【表 2 2】

ラベル	ビット数	説 明
読み出しイネーブル	1	DRAM からデータ読み出し
書き込みイネーブル	1	DRAM へデータ書き込み [ボックスモードでは無効]
パス X	1	入力 FIFO へ X(画素) 縦座標を渡す
パス Y	1	入力 FIFO へ Y(行) 縦座標を渡す
ループ	1	0 = データ中をループしない 1 = データ中をループする
予 備	11	必ず 0

20

30

読み出しイネーブル及び書き込みイネーブルに関する注意：

読み出しイネーブルがセットされたとき、I/O アドレス生成器は読み出しイタレータとして動作するので、特定の順序で画像を読み出し、画素を入力 F I F O に収容する。

【0 2 7 7】

書き込みイネーブルがセットされたとき、I/O アドレス生成器は書き込みイタレータとして動作するので、特定の順序で画像を書き込み、出力 F I F O から画素を取り出す。

【0 2 7 8】

読み出しイネーブルと書き込みイネーブルの両方がセットされたとき、I/O アドレス生成器は、読み出しイタレータ及び書き込みイタレータとして動作し、入力 F I F O 内の画素を読み出し、出力 F I F O からの画素を書き込む。画素は読み出された後に限り書き込まれる。即ち、書き込みイタレータは読み出しイタレータよりも先に進むことはない。このモードが使用されるときには、V L I W マイクロコードによる入力処理と出力処理の間のバランスを確保するように注意すべきである。尚、CacheGroup1 とキャッシュグループ 2 に異なる値をロードすることによって、別々のキャッシュグループを読み出しと書き込みに関して指定することも可能である。

40

【0 2 7 9】

PassX 及び PassY に関する注意：

PassX と PassY の両方がセットされた場合、Y 縦座標は X 縦座標よりも前に入力 F I F O に収容される。

【0 2 8 0】

50

PassX及びPassYは、読み出しイネーブルビットがクリアされたときに限りセットされる。縦座標をアドレス生成器に渡すのではなく、縦座標は入力F I F Oに直接収容される。縦座標はF I F Oから取り出されるときに前進する。

【0281】

書き込みイネーブルビットがセットされた場合、V L I Wプログラムは、入力F I F Oからの縦座標の読み出しを、出力F I F Oへの書き込みと確実にバランスさせる必要がある。なぜならば、書き込みは、縦座標までしか発生しないからである（読み出しイネーブル及び書き込みイネーブルに関する上記の注意を参照せよ。）

ループに関する注意：

ループビットがセットされた場合、読み出しは、一旦「最終画素、最終行」に達すると、「開始画素、開始行」で再開する。これは、データを繰り返し読み出す必要があるコンボリューションカーネルやディザセルマトリックスのような構造体を処理する場合に理想的である。

【0282】

読み出しイネーブル及び書き込みイネーブルがセットされた状態のループ処理は、単一ラインの履歴を維持する環境で役に立ち得るが、書き込みの前に読み出しが行われる場合に限り有効である。（長さに制約のある形式で書き込みが読み出しの前に行われる）F I F Oの効果がある場合、画像イタレータではなく、適切なテーブルI / Oアドレッシングモードが使用される。

【0283】

書き込みイネーブルだけがセットされたループ処理は、直前のN画素の書き込み窓を作成する。これは、窓からデータを読み出す非同期的プロセスを用いて使用される。A r t c a r d読み出しアルゴリズムは、このモードを利用する。

【0284】

順次読み出し及び書き込みイタレータ

図17は、画素データフォーマットの説明図である。最も簡単な画像イタレータは、順次読み出しイタレータ、及び対応した順次書き込みイタレータである。順次読み出しイタレータは、チャンネルからの画素を、同時に1行ずつ、上から下に示し、同じ行内では、画素は左から右へ示される。パディングバイトはクライアントに示されない。これは、画像からの各画素にある種のプロセスを実行する必要があるが、処理される画素の順序を問題にしないか、又はデータが特にこの順序であることを要求するアルゴリズムにとって最も有益な点である。順次読み出しイタレータを補完するのは、順次書き込みイタレータである。クライアントは出力F I F Oに画素を書き込む。順次書き込みイタレータは、適切なキャッシュ処理及び適切なパディングバイトを使用して、有効画像を書き出す。各順次イタレータは、2本のキャッシュラインへのアクセスを必要とする。読み出し時に、32画素が1本のキャッシュラインから示されるが、他のキャッシュラインはメモリからロードされ得る。書き込み時、32画素は1本のキャッシュラインに書き込まれるが、他の画素はメモリに書き込まれる。画像の各画素に独立に演算を実行するプロセスは、典型的に、画素を得るために順次読み出しイタレータを使用し、新しい画素値を目的画像内の対応した場所に書き込むため順次書き込みイタレータを使用するであろう。このような処理は図18に示されている。

【0285】

殆どの場合に、原画像及び目的画像は異なり、2個のI / Oアドレス生成器189と190によって表現される。しかし、原画像と目的画像を一致させることも有効である。なぜならば、ある種の入力画像は1回しか読み出されないからである。その場合、同じイタレータを入力と出力の両方に使用することが可能であり、読み出しイネーブルレジスタ及び書き込みイネーブルレジスタの両方が適切にセットされる。最大効率の場合、読み出し用と書き込み用の2個の異なるキャッシュグループを使用することが必要である。データがV L I Wプロセスによって作成され、順次書き込みイタレータによって書き込まれるべき場合、PassXフラグ及びPassYフラグは座標を生成するため使用され、その座標は、次に、

入力 F I F O へ伝えられる。V L I W プロセスは、これらの座標を使用し、出力データを適切に作成することができる。

【0286】

ボックス読み出しイタレータ

ボックス読み出しイタレータは、汎用フィルタ及びコンボルブのような演算を実行するため最も便利な順序で画素を提示するため使用される。このイタレータは、順次に読み出された画素の周りの正方形のボックス内の画素値を提示する。このボックスは、X 及び Y の方向の幅が 1、3、5 又は 7 画素になるように制限される（X ボックスサイズ及び Y ボックスサイズは同じ値であるか、一方の次元の値が 1 であり、他方の次元の値が 3、5 又は 7 である）。この処理は図 19 に示されている。

10

【0287】

ボックスオフセット：この専用レジスタは、どの入力画素がボックスの中心として使用されるかに関してサブサンプリングを決定するため使用される。通常値は 1 であり、これは、各画素がボックスの中心として使用されることを意味する。”2”という値は、画像ピラミッドを構築する場合のように画像を 4：1 に縮小する際に便利である。上記の図からの画素アドレスを使用すると、ボックスは、画素 0、2、8 及び 10 の順に中心が置かれる。ボックス読み出しイタレータは、最大で 14（2×7）本のキャッシュラインにアクセスする必要がある。画素が 7 本のラインの中の 1 組から提示される間に、他のキャッシュラインはメモリからロードすることが可能である。

20

【0288】

ボックス書き込みイタレータ

対応するボックス書き込みイタレータは存在しない。なぜならば、画素の複製は入力時だけに要求されるからである。入力用のボックス読み出しイタレータを使用するプロセスは、殆ど確実に、同期しているので出力用の順次書き込みイタレータを使用するであろう。その良い例はコンボルブである。コンボルブでは、N 個の入力画素が 1 個の出力画素を計算するため読み出される。そのプロセスフローは、図 20 に示されているようなフローである。ボックス読み出しイタレータを使用するとき、原画像及び目的画像は同じメモリを使ってはならない。なぜならば、後続の画像のラインが（新たに計算されたものではなく）元の値を必要とするからである。

30

【0289】

垂直ストリップ読み出し及び書き込みイタレータ

ある種の例では、画像を出力画素の順序で書き込む必要があるが、出力画素に対する入力画素のコヒーレンスの方向がわからない。その一例は回転である。画像が 90 度回転され、出力画像を水平方向に処理する場合、キャッシュコヒーレンスの完全に失われる。これに対して、出力画像を処理する同時に 1 本のキャッシュラインの幅の画素に基づいて出力画像を処理し、（同じライン上の次のキャッシュラインに相当する画素へ進むのではなく）次のラインへ進む場合、入力画像の画素に対するキャッシュコヒーレンスが得られる。入力画素に（色コヒーレンスのような）既知の「ブロック」コヒーレンスが存在する場合も同様であり、この場合、読み出しが処理順序を支配し、同期させられるべき書き込みは、同じ画素順序に従わなければならない。入力として提示された画素の順序（垂直ストリップ読み出し）、又は出力に予想される順序（垂直ストリップ書き込み）は同一である。順序は、ライン 0 の画素 0 から画素 31、次に、ライン 1 の画素 9 から画素 31、以下、画像の全てのラインに対して同様であり、次に、ライン 0 の画素 32 から 63 というように続く。最後の垂直ストリップでは、正確に 32 画素が存在しないかもしれない。この場合、画像の実際の画素だけが入力として、提示されるか、又は予定される。この処理は図 21 に示されている。

40

【0290】

垂直ストリップ書き込みイタレータだけを必要とするプロセスは、典型的に、出力画素座標が与えられた場合に入力画素座標をマッピングする方法を備えている。そのプロセスは、このマッピングに従って入力画像の画素にアクセスし、コヒーレンスは、入力画像用の

50

「ランダムアクセス」リーダー上に十分なキャッシュラインを設けることによって決定される。図 2 2 のプロセス概要に示されるように、座標は、典型的に、垂直ストリップ書き込みイタレータの PassX フラグ及び PassY フラグをセットすることによって生成される。

【0291】

書き込みイタレータを順次読み出しイタレータ又はボックス読み出しイタレータとペアにすることは重要ではないが、垂直ストリップ書き込みイタレータは、入力座標と出力座標の間に重要なマッピングが存在するとき、性能に著しい改良を加える。

【0292】

垂直ストリップ読み取りイタレータと垂直ストリップ書き込みイタレータをペアにすることは重要である。この場合、入力画像と出力画像が同じであるならば、垂直ストリップ読み取りイタレータと垂直ストリップ書き込みイタレータを一つの ALU188 に割り当てることが可能である。座標が必要であるならば、更なるイタレータをセットされた PassX フラグ及び PassY フラグと共に使用しなければならない。垂直ストリップ読み取り／書き込みイタレータは、入力 FIFO に画素を提示し、出力 FIFO からの出力画素を受け取る。適切なパディングバイトが書き込み時に挿入される。入力及び出力の各々は、優れた性能のために少なくとも 2 本のキャッシュラインを必要とする。

【0293】

テーブル I / O アドレッシングモード

(画像のような) テーブル内の値を調べることはしばしば必要になる。テーブル I / O アドレッシングモードは、この機能を提供し、クライアントがインデックスを出力 FIFO に入れることを要求する。I / O アドレス生成器は、インデックスを保有し、適切にデータを調べ、VLIW クライアントによる後の処理のため、調べた値を入力 FIFO に返す。

【0294】

1 次元、2 次元及び 3 次元のテーブルがサポートされ、特定のモードは補間を対象としている。VLIW のクライアント側の複雑さを軽減するため、インデックス値は、固定小数点の数値として取り扱われ、アクセス専用レジスタは、固定小数点を定め、インデックスの整数部として取り扱われるべきビットを定める。データフォーマットは、画素オフセットレジスタが無視される点で、汎用画像特徴の制限された形式であり、データは行内で連続していると仮定され、データ要素毎に、8 ビット又は 16 ビット (1 バイト又は 2 バイト) だけが許容される。4 ビットのアドレスモードレジスタは I / O タイプを決定するため使用される。

【0295】

【表 23】

ビット数	アドレスモード
3	1 = このアドレスモードはテーブル I/O である。
2 から 0	000 = 1 次元直接ルックアップ 001 = 1 次元補間(リニア) 010 = DRAM の FIFO 011 = 予備 100 = 2 次元補間(バイリニア) 101 = 予備 110 = 3 次元補間(トライリニア) 111 = 画像ピラミッドルックアップ

アクセス専用レジスタは次の通りである。

【0296】

【表 2 4】

レジスタ名	ローカル名	ビット数	説 明
アクセス専用 ₁	フラグ	8	読み出し及び書き込み用の汎用フラグ。より詳しい情報に関しては以下を参照のこと。
アクセス専用 ₂	フラクション X	8	X インデックスの小数ビットの個数
アクセス専用 ₃	フラクション Y	8	Y インデックスの小数ビットの個数
アクセス専用 ₄	フラクション Z	8	Z インデックスの小数ビットの個数
(下位 8 ビット／次の 12 ビット又は 24 ビット)	Z オフセット	12 又は 24	以下を参照のこと。

10

FractX、FractY及びFractZは、インデックスに基づいてアドレスを生成し、有意ビット及び整数／小数コンポーネントに関するインデックスのフォーマットを解釈するため使用される。種々のパラメータは、索引付けされたテーブルの次元数によって必要に応じて定められる。1次元テーブルは、FractXだけが必要であり、2次元テーブルは、FractX及びFractYを必要とする。各Fract値は、対応したインデックスの小数ビットの個数により構成される。例えば、Xインデックスは、5：3のフォーマットである。これは、整数が5ビットで、小数が3ビットであることを示す。したがって、FractXは3にセットされる。簡単な1次元ルックアップは、8：0のフォーマットであり、即ち、小数コンポーネントが全く無い。FractXは0になる。零オフセットは、3次元ルックアップの場合に限り必要になり、2種類の解釈を採用する。これは3次元テーブルルックアップのセクションに詳述されている。フラグレジスタ（アクセス専用1）は、データの読み取り（あるケースでは、書き込み）に影響を与える要因を決定するため使用される多数のフラグを含む。フラグレジスタは以下のように構成される。

20

【0 2 9 7】

【表 2 5】

ラベル	ビット数	説 明
読み取りイネーブル	1	DRAM からデータ読み取り
書き込みイネーブル	1	DRAM ヘデータ書き込み [1次元ルックアップの場合に限り有効]
データサイズ	1	0 = 8ビットデータ 1 = 16ビットデータ
予 備	5	必ず 0

30

1次元直接ルックアップとDRAMのFIFOを除くと、全てのテーブルI/Oモードは、読み出しだけをサポートし、書き込みをサポートしていない。したがって、上記の二つのモードを除く全てのI/Oモードでは、読み出しイネーブルビットはセットされ、書き込みイネーブルビットはクリアされる。1次元直接ルックアップは、読み出しイネーブルビットがセットされ、書き込みイネーブルビットがクリアされている読み出し専用と、読み出しイネーブルビットがクリアされ、書き込みイネーブルビットがクリアされている書き込み専用と、読み出しイネーブルビットと書き込みイネーブルビットの両方がセットされているリード・モディファイ・ライトと、の3モードをサポートする。

40

【0 2 9 8】

異なるモードは、以下の1次元ルックアップのセクションに記載されている。DRAM

50

F I F Oモードは、読み出しイネーブルビットと書き込みイネーブルビットの両方がセットされているライト・リード・モードという一つのモードだけをサポートする。

【0299】

このモードは、以下のD R A M F I F Oのセクションに記載されている。DataSizeフラグは、テーブルの各データ要素のサイズが8ビットであるか、16ビットであるかを決定する。2種類のデータサイズだけがサポートされる。32ビットの要素は、プロセスの必要条件に依存して、以下の2種類の方法の何れかで作成することができる：

二つの16ビットテーブルから同時に読み出し、その結果を合成する。これは、タイミングが問題となるときに都合が良いが、二つのI/Oアドレス生成器189と190を無駄に使用する点で欠点があり、各32ビット要素はC P Uによって32ビットのエンティティとして読むことができない；

16ビットテブから2回に亘って読み出し、その結果を合成する。これは、一つのルックアップだけが使用されるので都合が良いが、異なるインデックスを生成し、ルックアップへ伝えなければならない。

【0300】

1次元構造

直接ルックアップ

直接ルックアップは、1次元ルックアップテーブルへの簡単なインデキシングである。クライアントは、フラグレジスタに適切なビットをセットすることにより、三つのアクセスモード、即ち、

読み出し専用

書き込み専用

リード・モディファイ・ライト

の間で選択可能である。

【0301】

読み出し専用

クライアントは、固定小数点インデックスXを出力F I F Oに送り、テーブル[Int(X)]の8又は16ビット値が入力F I F Oに返される。インデックスXの小数コンポーネントは完全に無視される。インデックスが範囲外である場合、エッジ繰り返しフラグは、エッジ画素が返されるか、一定画素が返されるかを決定する。アドレス生成は、次のように簡単である：

データサイズが8ビットを示す場合、Xは右へバレルシフトされたFractXのビットであり、結果はテーブルのベースアドレスImageStartに加算される；

データサイズが16ビットを示す場合、Xは右へバレルシフトされたFractXのビットであり、左へ1ビットシフトされた（ビット0が0になる）結果は、テーブルのベースアドレスImageStartに加算される。

【0302】

結果として得られたアドレスにおける8又は16ビットデータ値は入力F I F Oに入れられる。アドレス生成には1サイクルを要し、要求されたデータをキャッシュから出力F I F Oへ転送するためにも（キャッシュヒットを仮定したとき）1サイクルを要する。例えば、各エントリーが16ビットであり、インデックスが12ビットの8：4の固定小数点フォーマットである256エントリーのテーブルの値を引く場合を想定すると、FractXは4であり、データサイズは1である。インデックスがルックアップへ渡されたとき、4ビットだけ右へシフトし、その結果を1ビットだけ左へシフトし、ImageStartに加算する。

【0303】

書き込み専用

クライアントは、固定小数点インデックスXを出力F I F Oへ渡し、その後、テーブルの指定された場所へ書き込まれるべき8又は16ビット値を渡す。完全な転送には最低でも

10

20

30

40

50

2 サイクル、即ち、アドレス生成のための 1 サイクルと、F I F O から D R A M へのデータ転送のための 1 サイクルを要する。V L I W プロセスがインデックスを F I F O に収容してから書き込まれるべき値を F I F O に収容するまでの間に任意の数のサイクル数を設けることができる。アドレス生成は、読み出し専用モードと同様に行われるが、データがそのアドレスから読み出されるのではなく、出力 F I F O からのデータがそのアドレスに書き込まれる。アドレスがテーブルの範囲外である場合、そのデータは F I F O から削除され、D R A M には書き込まれない。

【0304】

リード・モディファイ・ライト

クライアントは、固定小数点インデックス X を出力 F I F O に送り、テーブル[Int(X)]の 8 又は 16 ビット値が入力 F I F O へ返される。出力 F I F O に収容された次の値は、テーブル[Int(X)]に書き込まれ、前に返された値を置き換える。汎用処理ループは、ある場所から値を読み出し、値を変更し、その値を書き戻すプロセスである。全体的な時間は、インデックスからアドレスを生成するサイクルと、
テーブルからの値を返却するサイクルと、
ある種の方法で値を変更するサイクルと、
変更した値をテーブルへ書き戻すサイクルと、
の 4 サイクルである。

10

【0305】

クライアントが「Xからの読み出し」又は「Xへの書き込み」を表すフラグを提示する特定の読み出し／書き込みモードは存在しない。クライアントは、元の値を書き込むことにより「Xからの読み出し」のシミュレーションを行い、返された値をただ無視することによって「Xへの書き込み」のシミュレーションを行うことができる。しかし、このようなモードの使用が促進されることはない。なぜならば、各動作は、最低でも 3 サイクル（モディファイは必要ではない）を使用し、特定の読み出し及び書き込みモードによって行われるような 1 回のアクセスではなく 2 回のデータアクセスを使用するからである。

20

【0306】

補間テーブル

これは、所定の固定小数点インデックス X に対して、1 個の値ではなく 2 個の値が返される点を除くと、読み出しモードにおける直接ルックアップと同じである。返される値は、テーブル[Int(X)]とテーブル[Int(X)+1]である。いずれかのインデックスが範囲外である場合、エッジ繰り返しフラグは、エッジ画素が返されるか、一定画素が返されるかを決定する。アドレス生成は、8 ビットデータであるか 16 ビットデータであるかに応じて、2 番目のアドレスが単に（1 番目のアドレス + 1）又は（1 番目のアドレス + 2）である点を除くと、直接ルックアップと同じである。要求されたデータを出力 F I F O へ転送するためには 2 サイクルを要するが（キャッシュヒットを想定）、1 回の 16 ビットフェッチによって 2 個の 8 ビット値がキャッシュからアドレス生成器へ実際に返される。

30

【0307】

D R A M F I F O

読み出し／書き込み 1 次元テーブルの特殊なケースは、D R A M F I F O である。D R A M 及び付随したキャッシュを使用して所定の長さの F I F O をシミュレーションすることがしばしば必要である。D R A M の F I F O によって、クライアントは、テーブルに明示的に索引付けをせずに、あたかも F I F O の一方の端であるかのようにして出力 F I F O へ書き込み、同じ論理的 F I F O のもう一方の端であるかのようにして入力 F I F O から読み出す。2 個のカウンタがシミュレーティッド F I F O の入力位置と出力位置を追跡し、必要に応じて D R A M へキャッシュする。クライアントは、フラグレジスタの読み出しイネーブルビットと書き込みイネーブルビットの両方を見る必要がある。

40

【0308】

D R A M F I F O の用法の一例は、ある値のシングルライン履歴を保持することである。初期履歴は処理が始まる前に書き込まれる。一般的なプロセスがラインを完了するとき

50

、前のラインの値がF I F Oから獲得され、このラインの値はF I F Oに入れられる（このラインは、次のラインを処理するときには前のラインになる）。入力と出力が平均的に相互に一致する限り、出力F I F Oは常に一杯である。その結果として、この種のF I F Oの場合、アクセス遅延が無く効率的である（但し、F I F O全長が非常に短い場合、例えば、3又は4バイトの場合を除くが、その場合にはF I F Oの目的を損なうであろう）。

【0309】

2次元テーブル

直接ルックアップ

2次元直接ルックアップはサポートされていない。2次元ルックアップの全てのケースはバイリニア補間のためアクセスされることが予定されているので、特別なバイリニアルックアップが実現された。

10

【0310】

バイリニアルックアップ

この種のルックアップは、2次元テーブルからのデータのバイリニア補間のために必要である。固定小数点X座標及びY座標が与えられた場合（Y、Xの順に出力F I F Oに入れられた場合）、4個の値がルックアップ後に返される。それらの値は（順番に）：

Table[Int(X), Int(Y)]

Table[Int(X)+1, Int(Y)]

Table[Int(X), Int(Y)+1]

20

Table[Int(X)+1, Int(Y)+1]

である。

【0311】

返される値の順序は、最良のキャッシュコヒーレンスを与える。データが8ビットである場合、2個の値が2サイクルの間のサイクル毎に返され、下位バイトは第1のデータ要素である。データが16ビットである場合、1サイクル毎に1エントリーずつで、4個の値が4サイクルに返される。アドレス生成には2サイクルを要する。第1サイクルでは、FractYビットだけ右へバレルシフトされたインデックス（Y）がRowOffsetによって乗算され、その結果がImageStartに加えられる。第2サイクルは、XインデックスをFractXビットだけ右へシフトし、（8ビットデータの場合）その結果、又は（16ビットデータの場合）その結果を1ビットだけ左へシフトしたものの何れかが第1サイクルからの結果に加算される。これにより、アドレスAdrは、Table[Int(X),

30

Int(Y)]のアドレスである：

Adr = ImageStart

+ ShiftRight(Y, FractY)*RowOffset

+ ShiftRight(X, FractX)

Adrのコピーは後続のエントリーをフェッチするためAdrOldに保持される：

データが8ビットであるならば、タイミングは、アドレス生成の2サイクルであり、その後、データが返却される2サイクルが続く（1サイクル毎に2個のテーブルエントリー）；

40

データが16ビットであるならば、タイミングはアドレス生成の2サイクルであり、その後、データが返却される4サイクルが続く（1サイクル毎に1エントリー）。

【0312】

以下の二つの表は、8ビットデータサイズ及び16ビットデータサイズ用のアドレス計算方法を示している。

【0313】

【表 2 6】

サイクル	Adr から 2×8 ビットデータエントリーをフェッチする間に計算
1	$\text{Adr} = \text{Adr} + \text{RowOffset}$
2	<次のルックアップを準備>

【0 3 1 4】

【表 2 7】

サイクル	Adr から 1×16 ビットデータエントリーをフェッチする間に計算
1	$\text{Adr} = \text{Adr} + 2$
2	$\text{Adr} = \text{AdrOld} + \text{RowOffset}$
3	$\text{Adr} = \text{Adr} + 2$
4	<次のルックアップを準備>

10

両方のケースにおいて、アドレス生成の第 1 サイクルは、X インデックスの F I F O への挿入と重なってもよく、実効タイミングは、アドレス生成のための 1 サイクル、及び返却データの 4 サイクルと同程度に低い。インデックスの生成がその結果よりも 2 ステップ進んでいる場合、実効アドレス生成時間は無く、データは単に適切なレート（1 セット毎に 2 サイクル又は 4 サイクル）で生成される。

20

【0 3 1 5】

3 次元ルックアップ

直接ルックアップ

2 次元ルックアップの全てのケースはトライリニア補間のためアクセスされることが予想されるので、二つの特別なトライリニアルックアップが実現される。1 番目は単純なルックアップテーブルであり、2 番目は画像ピラミッドからのトライリニア補間用である。

【0 3 1 6】

トライリニアルックアップ

このタイプのルックアップは、色変換テーブルのような 3 次元データテーブルに有効である。標準的な画像パラメータは、データの単一の X Y 平面を定義し、即ち、各平面は、ImageHeight 個の行により構成され、各行は、RowOffset 個のバイトを含む。殆どの環境において、連続した平面を想定すると、一つの X Y 平面は、ImageHeight $\times$ RowOffset バイトの連なりである。このオフセットを仮定又は計算するのではなく、CPU によるソフトウェアは、そのオフセットを 12 ビット Zoffset レジスタの形で提供しなければならない。この形式のルックアップの場合、Z、Y、X の順の 3 個の固定小数点インデックスが与えら

30

Table[Int(X), Int(Y), Int(Z)]

Table[Int(X)+1, Int(Y), Int(Z)]

Table[Int(X), Int(Y)+1, Int(Z)]

Table[Int(X)+1, Int(Y)+1, Int(Z)]

40

Table[Int(X), Int(Y), Int(Z)+1]

Table[Int(X)+1, Int(Y), Int(Z)+1]

Table[Int(X), Int(Y)+1, Int(Z)+1]

Table[Int(X)+1, Int(Y)+1, Int(Z)+1]

が順番に返される。

【0 3 1 7】

返される値の順序は、最良のキャッシュコヒーレンスを与える。データが 8 ビットである場合、2 個の値が 4 サイクルの間のサイクル毎に返され、下位バイトは第 1 のデータ要素である。データが 16 ビットである場合、1 サイクル毎に 1 エントリーずつで、4 個の値が 8 サイクルに返される。アドレス生成には 3 サイクルを要する。第 1 サイクルでは、Fr

50

actZビットだけ右へバレルシフトされたインデックス（Z）が12ビットのZOffsetによって乗算され、ImageStartに加えられる。第2サイクルでは、FractYビットだけ右へバレルシフトされたインデックス（Y）がRowOffsetによって乗算され、その結果が前のサイクルの結果に加算される。第2サイクルは、XインデックスをFractXビットだけ右へシフトし、（8ビットデータの場合）その結果、又は（16ビットデータの場合）その結果を1ビットだけ左へシフトしたものの何れかが第2サイクルからの結果に加算される。これにより、アドレスAdrは、Table[Int(X),

Int(Y), Int(Z)]のアドレスである：

Adr = ImageStart

+ (ShiftRight(Z, FractZ)*ZOffset)

+ (ShiftRight(Y, FractY)*RowOffset)

+ ShiftRight(X, FractX)

Adrのコピーは後続のエントリーをフェッチするためAdrOldに保持される：

データが8ビットであるならば、タイミングは、アドレス生成の2サイクルであり、その後、データが返却される2サイクルが続く（1サイクル毎に2個のテーブルエントリー）；

データが16ビットであるならば、タイミングはアドレス生成の2サイクルであり、その後、データが返却される4サイクルが続く（1サイクル毎に1エントリー）。

【0318】

以下の二つの表は8ビットデータサイズ及び16ビットデータサイズ用のアドレス計算方法を示している。

【0319】

【表28】

サイクル	Adr から 2×8 ビットデータエントリーをフェッチする間に計算
1	Adr = Adr + RowOffset
2	Adr = AdrOld + ZOffset
3	Adr = Adr + RowOffset
4	<次のルックアップを準備>

【0320】

【表29】

サイクル	Adr から 1×16 ビットデータエントリーをフェッチする間に計算
1	Adr = Adr + 2
2	Adr = AdrOld + RowOffset
3	Adr = Adr + 2
4	Adr, AdrOld = AdrOld + ZOffset
5	Adr = Adr + 2
6	Adr = AdrOld + RowOffset
7	Adr = Adr + 2
8	<次のルックアップを準備>

両方のケースにおいて、アドレス生成のサイクルは、XインデックスのFIFOへの挿入と重なってもよく、単一の1回限りのルックアップの実効タイミングは、アドレス生成のための1サイクル、及び返却データの4サイクルと同程度に低い。インデックスの生成がその結果よりも2ステップ進んでいる場合、実効アドレス生成時間は無く、データは単に適切なレート（1セット毎に4サイクル又は8サイクル）で生成される。

【0321】

画像ピラミッドルックアップ

ブラッシング、タイリング及びワーピングの間に、画像の特定エリアの平均カラーを計

算することが必要である。所与のエリア毎に値を計算するのではなく、これらの機能は画像ピラミッドを利用する。画像ピラミッドの説明及び構造は、この文書のD R A Mインタフェース 8 1 の章の内部画像フォーマットに関するセクションに詳述される。このセクションは、順序付きの 3 個の固定小数点インデックス：レベル (Z)、Y、及び X に関して、ピラミッド内の所与の画素をアドレッシングする方法に関連する。画像ピラミッドルックアップは 8 ビットのデータエントリを想定するので、DataSize フラグは完全に無視される。Z、Y、及び X の指定後、以下の 8 画素：

[Int(X), Int(Y)] のレベル Int(Z) の画素
 [Int(X)+1, Int(Y)] のレベル Int(Z) の画素
 [Int(X), Int(Y)+1] のレベル Int(Z) の画素
 [Int(X)+1, Int(Y)+1] のレベル Int(Z) の画素
 [Int(X), Int(Y)] のレベル Int(Z)+1 の画素
 [Int(X)+1, Int(Y)] のレベル Int(Z)+1 の画素
 [Int(X), Int(Y)+1] のレベル Int(Z)+1 の画素
 [Int(X)+1, Int(Y)+1] のレベル Int(Z)+1 の画素
 が入力 F I F O を介して返される。

10

【 0 3 2 2 】

8 個の画素は 4×16 ビットのエントリとして返却され、X エントリと X + 1 エントリはハイ／ローに合成される。例えば、拡大縮小された (X, Y) 座標が (10.4, 12.7) であるならば、最初に返される 4 個の画素は、(10, 12)、(11, 12)、(10, 13) 及び (11, 13) であろう。座標が有効レンジ外であるとき、クライアントは、エッジ画素繰り返し及び定数画素レジスタ（下位 8 ビットだけが使用される）によって、エッジ画素繰り返し、又は定数カラー値の返却を選択することができる。画像ピラミッドが構築されたとき、レベル 0 の座標からレベル Z の座標までの簡単なマッピングが存在する。この方法は、単に、X 又は Y 座標を Z ビットだけ右へシフトさせる。これは、座標の整数部を獲得するため既にシフトされたビット数（即ち、X 座標に対する FractX ビットの右シフト及び Y 座標に対する FractY の右シフト）に加えて行う必要がある。画像ピラミッドの所与のレベルに対する ImageStart 及び RowOffset を見つけるため、24 ビットの ZOffset レジスタがレベル情報テーブルへのポインタとして使用される。このテーブルは、レコードの配列であり、各レコードは、レベル数によって順序付けられた所与のピラミッドのレベルを表現する。各レコードは、ImageStart からそのピラミッドのレベルまでの 16 ビットのオフセット ZOffset（オフセットの下位 6 ビットのような 64 バイトの整列したアドレスは存在しない）と、そのレベルのための 12 ビットの ZRowOffset と、により構成される。テーブルの要素 0 は、ZOffset が 0 であり、単にフルサイズ画像を指示するとき、ZRowOffset が汎用レジスタ RowOffset と一致する。テーブルの要素 N における ZOffset 値は、画像ピラミッドのレベル N の実効 ImageStart を得るため、ImageStart に加算されるであろう。テーブルの要素 N の RowOffset 値は、レベル N に対する RowOffset 値を格納する。C P U 上で動かされるソフトウェアは、このアドレッシングモードを使用する前にテーブルを適切にセットアップしなければならない。実際のアドレス生成の概要は、以下で 1 サイクル毎に説明される。

20

30

40

【 0 3 2 3 】

【表 3 0】

サイクル	ロードレジスタ	始まりのアドレス	他の演算
0	—	—	$Z\text{Adr} = \text{ShiftRight}(Z, \text{Frac}Z) + Z\text{Offset}$ $Z\text{Int} = \text{ShiftRight}(Z, \text{Frac}Z)$
1	Zoffset	Zadr	$Z\text{Adr} += 2$ $Y\text{Int} = \text{ShiftRight}(Y, \text{Frac}Y)$
2	ZrowOffset	ZAdr	$Z\text{Adr} += 2$ $Y\text{Int} = \text{ShiftRight}(Y\text{Int}, Z\text{Int})$ $\text{Adr} = Z\text{Offset} + \text{ImageStart}$
3	Zoffset	ZAdr	$Z\text{Adr} += 2$ $\text{Adr} += Z\text{rowOffset} * Y\text{Int}$ $X\text{Int} = \text{ShiftRight}(X, \text{Frac}X)$
4	Zadr	ZAdr	$\text{Adr} += \text{ShiftRight}(X\text{Int}, Z\text{Int})$ $Z\text{Offset} += \text{ShiftRight}(X\text{Int}, 1)$
5	FIFO	Adr	$\text{Adr} += Z\text{rowOffset}$ $Z\text{Offset} += \text{ImageStart}$
6	FIFO	Adr	$\text{Adr} = (Z\text{Adr} * \text{ShiftRight}(Y\text{Int}, 1)) + Z\text{Offset}$
7	FIFO	Adr	$\text{Adr} += Z\text{adr}$
8	FIFO	Adr	<次の検索のためのサイクル 0>

10

20

上述のアドレス生成は、単一のバレルシフタ、2 個のアダー、及び 24 ビットを生じる単一の 16×16 乗算／加算ユニットを使用して実現できる。一部のサイクルには 2 回のシフトが含まれるが、それらは、シフト値が同じであるか（即ち、バレルシフタの出力が 2 回使用されるか）、又はシフトが 1 ビットであるかのどちらかであり、ハードワイヤード可能である。次の内部レジスタ：ZAdr, Adr, ZInt, YInt, XInt, ZRowOffset, ZImageStart が必要である。_Int 形のレジスタは最大で 8 ビットだけが必要であるが、他のレジスタは最大 24 ビットになり得る。このアクセス方法は、画像ピラミッドから読み出すだけであり、画像ピラミッドへ書き込まないので、キャッシュグループ 2 (CacheGroup2) が (Zadr を用いて) 画像ピラミッドアドレステーブルをルックアップするため使用される。CacheGroup1 は、(Adr を用いて) 画像ピラミッド自体をルックアップするため使用される。アドレステーブルは、(原画像サイズに応じて) 約 22 エントリーがあり、各エントリーは 4 バイトである。したがって、3 又は 4 本のキャッシュラインをキャッシュグループ 2 に割り付けるべきであり、できるだけ多数のキャッシュラインを CacheGroup1 に割り付けるべきである。データのセットを返すためのタイミングは 8 サイクルであり、サイクル 8 とサイクル 0 は動作が重なり合うと考えられ、即ち、次の要求のサイクル 0 はサイクル 8 の間に生じる。これが許される理由は、サイクル 0 がメモリにアクセスせず、サイクル 8 が特定の演算を含まないからである。

30

【0324】

40

VLIW ベクトルプロセッサ 74 を用いる座標の生成

書き込みイタレータにリンクされた一部の関数は、処理パイプラインの一部で処理されている現在画素の X 及び／又は Y 座標を必要とする。特定の処理は、処理されている各行又は列の最後に行われる必要がある。殆どの場合に、PassX 及び PassY フラグは全ての座標を完全に生成するために十分である。しかし、特別な必要条件が存在する場合、以下の関数を使用することができる。この計算は、単一サイクル生成のため多数の ALU に展開することが可能であり、或いは、多数サイクル生成のため単一の ALU 188 内で行ってもよい。

【0325】

シーケンシャル [X, Y] 生成

50

プロセスが順次読み出しイタレータに従って順次に画素を処理するとき（又は、画素を生成し、順次書き込みイタレータへその画素を書き出すとき）、以下のプロセスは、図 2 3 に示されるように、PassXフラグ／PassYフラグの代わりに、X、Y座標を生成するため使用可能である。

【0326】

座標生成器は、X座標に関して画像幅ImageWidthまでカウントアップし、ImageWidth個の画素毎に1回ずつY座標をインクリメントする。実際のプロセスは図 2 4 に示され、以下の定数がソフトウェアによってセットされる。

【0327】

【表 3 1】

10

定 数	値
K_1	ImageWidth
K_2	ImageHight(オプション)

以下のレジスタは一時変数を保持するため使用される。

【0328】

【表 3 2】

20

変 数	値
Reg ₁	X (各ラインで 0 からスタート)
Reg ₂	Y (0 からスタート)

必要条件は以下のように要約される。

【0329】

【表 3 3】

30

必要条件	*+	+	R	K	LU	イタレータ
一 般	0	3/4	2	1/2	0	0
合 計	0	3/4	2	1/2	0	0

垂直ストリップ [X, Y] 生成

プロセスが、画素を垂直ストリップ書き込みイタレータへ書くために画素を処理し、ある種の理由からPassXフラグ／PassYフラグを使用できないとき、図 2 5 に示されるようなプロセスがX、Y座標を生成するため使用できる。この座標生成器は、X座標に関して画像幅ImageWidthまでカウントアップし、ImageWidth個の画素毎に1回ずつY座標をインクリメントするだけである。実際のプロセスは図 2 6 に示され、以下の定数がソフトウェアによってセットされる。

【0330】

40

【表 3 4】

定 数	値
K_1	32
K_2	ImageWidth
K_3	ImageHight

以下のレジスタは一時変数を保持するため使用される。

【0 3 3 1】

【表 3 5】

変 数	値
Reg ₁	StartX (0 からスタート、垂直ストリップ毎に 1 回ずつ 32 だけインクリメントされる)
Reg ₂	X
Reg ₃	EndX (32 からスタート、垂直ストリップ毎に 1 回ずつ ImageWidth の最大値まで 32 だけインクリメントされる)
Reg ₄	Y

必要条件は以下のように要約される。

【0 3 3 2】

【表 3 6】

必要条件	*+	+	R	K	LU	イタレータ
一 般	0	4	4	3	0	0
合 計	0	4	4	3	0	0

垂直ストリップ毎に 1 回ずつ出現する計算（2 回の加算、そのうち一つは M I N 演算と関連付けられている）は、一般的なタイミング統計量に含まれない。なぜならば、それらは、画素タイミング毎の実測の一部ではないからである。しかし、それらは、特定の関数用のマイクロコードのプログラミングに考慮しなければならない。

【0 3 3 3】

イメージセンサインタフェース（I S I 8 3）

イメージセンサインタフェース（I S I 8 3）は、C M O S イメージセンサからデータを取得し、データを D R A M に蓄積するため利用できるようにする。イメージセンサは、3 : 2 のアスペクト比をもち、典型的な解像度は 7 5 0 × 5 0 0 サンプルであり、3 7 5 K（1 画素当たり 8 ビット）を生ずる。2 × 2 画素ブロックの各々は、図 2 7 に示されるような構造を有する。I S I 8 3 は、画像を読み出すため、フレーム同期パルス及び Pixel Clock パルスを含む制御情報をイメージセンサへ送信する状態機械である。画素は、イメージセンサから読み出され、V L I W 入力 F I F O 7 8 に入れられる。V L I W は、次に、画素を処理及び／又は蓄積することが可能である。これは図 2 8 に示されている。I S I 8 3 は、検出された写真画像を D R A M に蓄積する V L I W プログラムと組み合わせて使用される。処理は次の 2 ステップで行われる：

小規模 V L I W プログラムは F I F O から画素を読み出し、順次書き込みイタレータを介して画素を D R A M へ書き込む；

D R A M 内の写真画像は、写真が撮影されたときのカメラの向きに応じて 9 0、1 8 0 又は 2 7 0 度回転させられる。

【0334】

回転が0度である場合、ステップ1は、写真画像を最終的な写真画像格納場所へ書き出すだけであり、ステップ2は実行されない。回転が0度以外である場合、画像は一時エリア（例えば、プリント画像記憶エリア）に書き込まれ、次に、ステップ2の間に、回転させられ最終的な写真画像格納場所へ入れられる。ステップ1は、非常に簡単なマイクロコードであり、V L I W入力F I F O 7 8からデータを取り出し、そのデータを順次書き込みイタレータに書き込む。ステップ2の回転は、加速V a r kアフィン変換関数を用いて実行される。処理は、設計の複雑さを軽減し、画像のために既に必要とされたV a r kアフィン変換回転ロジックを再使用するため、2ステップで行われる。これが許容できる理由は、両方のステップが、A r t c a mのオペレータにとってごくわずかな時間である約0.03秒内に終了するからである。たとえそうであったとしても、読み出しプロセスは、センサ速度による制限を受け、フレーム全体を読み出すために0.02秒を要し、画像を回転するため約0.01秒を要する。

10

【0335】

検出された写真画像と内部フォーマット画像との間で変換を行うために向きは重要である。なぜならば、R画素、G画素及びB画素の相対位置は向きと共に変化するからである。処理された画像は、印刷のため正しい向きにするために、印刷プロセス中に回転させる必要がある。A r t c a mの3次元モデルは、2個のイメージセンサを具備し、それらの入力は単一のI S I 8 3（マイクロコードは異なるが、A C P 3 1は同一）に多重化される。各センサはフレーム蓄積型であるため、両方の画像を同時に取得し、同時にメモリへ転送することができる。

20

【0336】

ディスプレイコントローラ88

A r t c a m上の「撮影」ボタンが半押しされたとき、T F Tはイメージセンサからの（簡単なV L I Wプロセスによって変換された）現在の像を表示する。撮影ボタンが完全に押下されたとき、撮影像が表示される。ユーザが印刷ボタンを押下し、画像処理が始まると、T F Tはターンオフされる。画像が印刷されると、T F Tは再度ターンオンされる。ディスプレイコントローラ88は、フラットパネルディスプレイを内蔵するA r t c a mモデルで使用される。例示的なディスプレイは、240×160画素の解像度を有するT F T型L C Dである。ディスプレイコントローラ88の構成は図29に示されている。ディスプレイコントローラ88の状態機械はレジスタを含み、レジスタは同期生成のタイミングを制御し、表示画像はそこから（特定のキャッシュグループによるデータキャッシュ76を介してD R A Mに）取得され、レジスタは、T F Tをその時点で（T F Tイネーブルを用いて）動作可能状態にすべきであるかどうかを制御する。C P Uは、低速バスを介してこれらのレジスタへ書き込むことが可能である。240×160画素の画像をR G B方式T F Tに表示するためには、1画素当たり3個のコンポーネントが必要になる。D R A Mから取得された画像は3個のD A Cを介して表示され、D A Cは、R、G及びBの出力信号の各々に対して1個ずつ設けられる。毎秒30フレーム（毎秒60フィールド）の画像リフレッシュレートで、ディスプレイコントローラ88は、 $240 \times 160 \times 3 \times 30 = 3.5 \text{ MB/秒}$ のデータ転送速度を必要とする。

30

40

【0337】

このデータ速度は、システムの他の部分よりも低速である。しかし、集約的な画像処理中にV L I Wプログラムを減速させるためには十分に高い。T F T動作の一般的な原理はこれを反映している。

【0338】

画像データフォーマット

上述の通り、D R A Mインタフェース81は、A C Pチップの他のクライアント部と、R A M B U S

D R A Mとの間を接続する。事実上、D R A Mインタフェース内の各モジュールはアドレ

50

ス生成器である：

A C Pによって操作される画像には3種類の論理的タイプがある。それらは、
 - C C Dから捕捉された入力像であるC C D像
 - A r t c a m装置によって内部的に利用される内部画像フォーマット
 - A r t c a mによって印刷される出力画像フォーマットである印刷画像
 である。

【0339】

これらの画像は、典型的に、色空間、解像度、並びに、カメラ間で変動し得る出力及び入力色空間が相違する。例えば、低価格カメラのC C D像は、高価格カメラで使用されるC C Dとは、解像度やカラー特性が異なる。しかし、全ての内部画像フォーマットは、あらゆるカメラの中で色空間に関して同一フォーマットである。

10

【0340】

その上、3種類の画像タイプは、どちらの方向が「上向き」であるかに関して異なる可能性がある。カメラの物理的な向きは、人物像であるか、又は風景像であるかについての観念を生じさせ、この観念は処理全体を通じて維持されるべきである。このため、内部画像は常に正しく向きを決められ、印刷動作中にC C Dから獲得された画像に対して回転が実行される。

【0341】

C C D像編成

多種多様なC C Dイメージセンサを利用可能であるが、C C D自体は750×500形イメージセンサであり、375000バイト（1画素当たり8ビット）を生成する場合を想定する。2×2画素ブロックの各々は図30に示されるような構造を有する。

20

【0342】

D R A Mに蓄積されたときのC C D像は、メモリ内で隣接している所定のラインと連続した画素を有する。各ラインは、次々に蓄積される。イメージセンサインタフェース83は、C C Dからデータを取得し、そのデータを正しい向きでD R A Mに格納する役割を担う。このようにして、回転0度のC C D像は、G、R、G、R、G、R、...のような第1ラインを含み、B、G、B、G、B、G、...のような第2ラインを含む。C C D像が人物像であり、90度回転されているならば、第1ラインはR、G、R、G、R、Gであり、第2ラインはG、B、G、B、G、B、...（以下同様に続く）となるであろう。

30

【0343】

画素は、インターリーブ形式で蓄積される。なぜならば、全てのカラーコンポーネントが内部画像フォーマットを変換するために必要になるからである。

【0344】

A C P 3 1はC C D画素フォーマットに関して仮説を立てないことに注意する必要がある。なぜならば、イメージング用の実際のC C Dは、A r t c a mの間で経時的に変化し得るからである。ハードウェアによって行われる全ての処理は、A C P 3 1の有用性を拡大するために主要なマイクロコードによって制御される。

【0345】

内部画像編成

内部画像は、典型的に、多数のチャンネルにより構成される。V a r k画像には、

40

L a b

L a b α

L a b Δ

$\alpha \Delta$

L

が含まれるが、これらに限定されるものではない。

【0346】

L、a及びbは、L a b色空間のコンポーネントに対応し、 α は（コンポジットのため使用される）マツトチャンネルであり、 Δ は（ブラッシング、タイリング、及びイルミネー

50

ティングの間に使用される) バンプマップチャンネルである。

【0347】

VLIWプロセッサ74は、画像が平面的な構造で編成されることを要求する。そのため、Lab画像は、三つの別個のメモリブロック、即ち、

Lチャンネル用の1ブロック

aチャンネル用の1ブロック

bチャンネル用の1ブロック

として記憶される。

【0348】

各チャンネルのブロック内において、画素は(幾つかのオプションのパディングバイトと共に) 所定の行に対し隣接して記憶され、行は次々に記憶される。

10

【0349】

図31には、論理的画像100の記憶形式の一例が示されている。論理的画像100は、逐次記憶されたカラーコンポーネントL101、a102及びb103を有する平面形式で記憶される。或いは、論理的画像100は、非圧縮Lコンポーネント101と、圧縮Aコンポーネント105及び圧縮Bコンポーネント106を有する圧縮フォーマットで記憶してもよい。

【0350】

図32を参照すると、ラインn(110)の画素は、ラインn+1(111)の画素の前に一緒に格納されている。画像は単一チャンネル内の隣接したメモリに記憶されている。

20

【0351】

8MBメモリモデルの場合、全処理の終了後の最終的な印刷画像は、クロミナンスチャンネルにおいて圧縮する必要がある。クロミナンスチャンネルの圧縮は、4:1にすることができ、全体的な圧縮は12:6又は2:1になる。

【0352】

最終的な印刷画像を除くと、Artcamの画像は典型的に圧縮されない。メモリ制約のため、ソフトウェアは、クロミナンスチャンネルの最終的な印刷画像を、これらのチャンネルの各々を2:1でスケーリングすることによって圧縮することを選択してもよい。これが行われた後、画像を印刷するために利用されるプリントVark関数は、圧縮されたものとして指定されたクロミナンスチャンネルを取り扱うように指示されるべきである。プリント関数は、圧縮されたクロミナンスを取り扱う方法を知っている唯一の関数であるが、たとえそうであるとしても、プリント関数は、固定の2:1圧縮比で処理するだけである。

30

【0353】

画像を圧縮し、次に、最終印刷画像を作成するため圧縮画像に基づいて動作することは可能であるが、これは、解像度の損失を生ずるので推奨できない。その上、画像は、プリントアウト前の最終段階として一回だけ圧縮されるべきである。1回の圧縮は実質的には気付かれないのに対し、多数回の圧縮はかなりの画像劣化を生じさせる。

【0354】

クリップ画像編成

40

Artcardに記憶されたクリップ画像はACP31によって明示的にはサポートされない。ソフトウェアは、現在のArtcardから画像を取り出し、ACPで認識できる形式にデータを編成する役割を担う。画像がArtcard上に圧縮形式で記憶されているならば、ソフトウェアは、それらを伸長する役割を担う。なぜならば、Artcard画像の伸長をサポートする特定のハードウェアは存在しないからである。

【0355】

画像ピラミッド編成

画像を操作するため利用されるブラッシング、タイリング、及びワーピングプロセス中に、しばしば、画像の特定のエリアの平均カラーを計算することが必要である。所与の各エリアの値を計算するのではなく、これらの関数は、画像ピラミッドを利用する。図33に

50

示されるように、画像ピラミッドは、事実上の多重解像度画素マップである。原画像 115 は 1 : 1 表現である。各次元におけるローパスフィルタリング及び 2 : 1 のサブサンプリングによって、元のサイズの $1/4$ の画像 116 が得られる。このプロセスは、画像全体が単一画素によって表現されるまで続けられる。画像ピラミッドは、原始的な内部フォーマット画像から構築され、原画像が占めていたサイズの $1/3$ ($1/4 + 1/16 + 1/64 + \dots$) を使用する。1500 × 1000 の原画像の場合、対応した画像ピラミッドは約 $1/2$ MB である。画像ピラミッドは特定の V a r k 関数によって構築され、他の V a r k 関数のパラメータとして使用される。

【0356】

印刷画像編成

10

処理画像全体は、それを印刷するため同時に必要になる。しかし、印刷画像出力は、CMY ディザ画像を含み、唯一の一時的な画像フォーマットであり、印刷画像機能内で使用される。しかし、内部色空間から印刷色空間への変換が必要であることに注意しなければならない。その上、色変換は、インク特性の異なるカメラのプリントロールの種類毎に別々に調整することができる。例えば、セピア出力は特定のセピア調色 A r t c a r d を使用して実行され、又はセピア色調プリントロールを使用して実行される（したがって、全ての A r t c a r d はセピア色調で機能する）。

【0357】

色空間

上述の通り、A r t c a m では、画像タイプの種類に対応して三つの色空間が使用される。

20

【0358】

A C P は、特定の色空間についての直接的な知識を持たない。その代わりに、A C P は、C C D 色空間、内部色空間、及びプリンタ色空間、即ち、

C C D 色空間：R G B と、

内部色空間：L a b と、

プリンタ色空間：C M Y と、

の間の変換のためクライアントの色空間変換テーブルに依存する；

A C P 31 から色空間変換を取り除くことによって、

— 様々なカメラで様々な C C D を使用すること

30

— 同じカメラで（長期間に異なるプリントロールの）異なるインクを使用すること

— A C P 設計パスから C C D 選定を分離すること

— 正確なカラー処理のため内部色空間を巧く定義すること

が可能になる。

【0359】

A r t c a r d インタフェース 87

A r t c a r d インタフェース (A I) は、A r t c a r d がリニアイメージセンサの下を通過するとき、リニアイメージセンサからデータを取得し、どのデータを D R A M に記憶するため利用する。イメージセンサは、1 走査線あたりに 11000 個の 8 ビットサンプルを生成し、A r t c a r d を 4800 d p i でサンプリングする。A I は、画像を読み出すため、LineSync パルス及び PixelClock パルスを含む制御情報をリニアセンサへ送る状態機械である。画素は、リニアセンサから読み出され、V L I W 入力 F I F O 78 に収容される。V L I W は、次に、画素を処理する、及び／又は、画素を蓄積することが可能である。A I は以下の数個のレジスタを含む。

40

【0360】

【表 3 7】

レジスタ名	説 明
画素数	センサライン内の画素数(約 11, 000)
状 態	プリントヘッドインタフェースの状態レジスタ
残り画素数	現在ラインの残りバイト数
アクション	
リセット	このレジスタへの書き込みは、AI をリセットし、全てのスキヤニングを停止し、全レジスタに 0 をロードする。
スキャン	このレジスタへの非零値の書き込みは、状態レジスタのスキヤニングビットをセットし、Artcard インタフェース走査サイクルを開始させる。 このレジスタへの 0 の書き込みは、スキヤニングプロセスを停止し、状態レジスタのスキヤニングビットをクリアする。 走査サイクルは、全ての AI にセンサから VLIW 入力 FIFO78 へ画素数個のバイトを転送させ、適切に画素クロック信号を生成する。画素数個のバイトの完了後、ライン同期パルスが与えられ、同期サイクルが再開する。 残り画素数レジスタは現在の走査ラインに読み出されずに残っている画素数を保持する。

10

20

尚、CPU は、走査を始動する前に VLIW 入力 FIFO78 をクリアしなければならない。状態レジスタは以下のように解釈されるビットを有する。

【0 3 6 1】

【表 3 8】

ビット名	ビット	説 明
スキヤニング	1	セットされているならば、AI はスキヤニング中であり、現在ラインから転送されるべき残り画素数は残り画素数レジスタに記録されている。 クリアされているならば、AI はスキヤニング中ではなく、画素を VLIW 入力 FIFO78 へ転送していない。

30

40

Artcard インタフェース (AI) 87

Artcard インタフェース (AI) 87 は、Ardcard リーダー 34 から Artcard 画像を取得し、それをオリジナルデータ (通常は、Vark スクリプト) に復号化する役割を担う。特に、AI 87 は、Artcard スキャナリニア CCD 34 から信号を受け取り、カードに印刷されたビットパターンを検出し、ビットパターンをオリジナルデータに変換し、読み出しエラーを訂正する。

【0 3 6 2】

Artcard 9 が挿入されていない場合、Artcam から印刷された画像は、任意の標準的な画像処理ルーチンによって整理された検出写真画像に過ぎない。Artcard 9 は、ユーザがプリントアウトする前に写真を修正するための手段である。特定の Artcard 9 を Artcam に挿入するという簡単なタスクによって、ユーザは、写真画像に対して実行される複雑な画像処理を定義することができる。

【0 3 6 3】

Artcard が挿入されていない場合、写真画像は、印刷画像を作成するため通常の方法で処理される。1 枚の Artcard 9 が Artcam に挿入されているとき、その Artcard の効果は、印刷画像を生成するため写真画像に適用される。Artcard

50

9が取り外されたとき（取り出されたとき）、印刷された画像は標準的な方法で処理された写真画像に戻る。ユーザがArt cardを取り出すためボタンを押下したとき、Art cam中央プロセッサ31で実行されるオペレーティングシステムによって維持されるイベントキューにイベントが収容される。イベントが処理されるとき（例えば、現在のプリントが行われた後）、以下の状況が発生する。

【0364】

現在のArt cardが有効であるならば、印刷画像は無効としてマークされ、「プロセス標準」イベントがイベントキューに入れられる。イベントが最終的に処理されたとき、印刷画像を生成するため、写真画像に標準的な画像処理演算が加えられる。

【0365】

モーターはArt cardを取り出すため始動され、時間的に規定された「モーター停止」イベントがイベントキューに追加される。

【0366】

アートカードの挿入

ユーザがArt card 9を挿入したとき、Art card センサ49はそれを検知し、ACP 72へ報せる。これにより、ソフトウェアは、「Art card 挿入済み」イベントをイベントキューに挿入する。このイベントが処理されると、以下の数個の状況が発生する：

現在のArt cardは無効としてマークされる（「無し」ではない）；

印刷画像は無効としてマークされる；

Art card モーター37はArt card を取り込むため始動される；

Art card インタフェース87はArt card を読み取るように命令される；

Art card インタフェース87は、Art card スキャナリニアCCD 34から信号を受け取り、カードに印刷されたビットパターンを検出し、検出されたビットパターンの誤りを訂正し、有効Art card データブロックをDRAMに生成する。

【0367】

Art card CCDからのデータ読み出し—一般的考察

図34に示されるように、データカード読み出しプロセスは、画素データがカードから読み出される間に4フェーズを有する。フェーズは以下の通りである：

フェーズ1 Art card 上のデータエリア検出

フェーズ2 CCD画素に基づくArt card からのビットパターン検出と、バイトとしての書き込み

フェーズ3 バイトパターンのスクランブル解除及びXOR演算

フェーズ4 データ復号化（リード・ソロモン復号）。

【0368】

図35に示されるように、Art card 9は、ナイキストの定理を充足するため、少なくとも印刷された解像度の2倍でサンプリングされなければならない。実際には、これよりも高いレートでサンプリングする方がよい。好ましくは、画素は、各次元で印刷されたドットの解像度の3倍でサンプリングされ（237）、1個のドットを定義するために9個の画素が必要になる。したがって、Art card 9の解像度が1600 dpiであり、センサ34の解像度が4800 dpiである場合、50mmのCCDイメージセンサを使用することにより、1列毎に9450画素が得られる。即ち、2MBのドットデータ（1ドット当たり9画素の場合）が必要であるならば、 $2\text{MB} \times 8 \times 9 / 9450 = 15978$ 列＝約16000列が必要になる。勿論、ドットがサンプリングCCDと厳密に揃えられていないならば、考えられる最悪のケースでは、ドットは16（4×4）画素エリア（231）で検出されるであろう。

【0369】

Art card 9は、熱損傷のため多少歪みが生じ、Art card リーダーへの挿入のばらつきによって僅かに（最大で、例えば、1度）回転し、リーダーモーター37の速度の変動を原因として真のデータレートに僅かな偏差が現れる可能性がある。これらの変化

10

20

30

40

50

のために、カードからのデータの列は、対応した画素データの列として読み出せなくなる。図36に示されるように、Art card 9の1度の回転によって、カード上のある列からの画素は、166列に亘る画素として読み出されることになる。

【0370】

最後に、Art card 9は、人間のオペレータにとって合理的な時間内に読み取られなければならない。Art card上のデータはArt card表面の大部分を覆っているので、タイミングの問題はArt cardデータ自体に制限され得る。1.5秒の読み出し時間は、Art card読み出しとして適切である。

【0371】

Art cardは1.5秒内に取り込まれなければならない。したがって、16000列の全ての画素データがCCD34から1.5秒以内に、即ち、1秒当たり10667列で読み出される必要がある。したがって、1列を読み取るために利用できる時間は1/10667秒、即ち、93747ナノ秒である。画素データは画素データを読み出すプロセスとは完全に独立に、同時に1列ずつDRAMへ書き込むことが可能である。

【0372】

1列のデータ（1画素当たり4ビット、したがって、1バイト当たり2×4ビット画素の読み出しが可能であるので、9450/2バイト）をDRAMへ書き込むための時間は、8個のキャッシュラインを使用することによって短縮される。4ラインが同時に書き込まれるならば、4バンクを独立にかきこむことが可能であり、オーバーラップレイテンシーが短縮される。この場合、4725バイトは11840ns（4725/128×320ns）で書き込み可能である。このようにして、所定の列データをDRAMへ書き込むために要する時間は、利用可能な帯域幅の13%未満しか使用しない。

【0373】

Art cardの復号化

単純にデータサイズを見ると、リニアCCD34によって読み取られるようなArt card画素データ全体（各ビットが3×3配列として読まれるならば140MB）が維持される場合、プロセスを8MBのメモリ33に適合させることは不可能に思われる。このため、リニアCCDの読み取り、ビットマップの復号化、及びアンビットマッププロセスは、（Art card 9がリニアCCD34の傍を通過する間に）リアルタイムで行われるべきであり、これらのプロセスは、全ての画像データ蓄積が利用可能にされなくても効率的に作用しなければならない。

【0374】

Art card 9が挿入されたとき、古い蓄積された印刷画像と拡大された写真画像は無効になる。新しいArt card 9は、現在の撮影写真画像に基づいて新しい画像を作成する命令を含み得る。古い印刷画像は無効であり、拡大された写真画像及び画像ピラミッドを保持するエリアは無効であり、読み出しプロセス中にスクラッチメモリとして使用できる5MBよりも多くの量が残される。厳密に説明すると、Art cardの新しい生データが書き込まれるべき1MBのエリアは、最終的なリード・ソロモン復号化が行われるときまで、Art card読み出しプロセス中にスクラッチデータとして使用可能であり、その1MBエリアが再度開放される。ここで説明している読み出しプロセスは、（データの最終的な宛先としての利用を除いて）余分な1MBエリアを利用しない。

【0375】

また、スクランブル解除プロセスは、スクランブル解除を所定の場所で行うことができないので、2組の2MBのメモリエリアを必要とすることに注意すべきである。幸運にも、5MBのスクラッチエリアはこの目的のために十分なスペースを含む。

【0376】

図37を参照すると、Art cardデータを復号化するために必要なステップのフローチャート220が示されている。これらのステップには、Art card 221を読み込むステップと、対応した符号化され、XOR演算され、スクランブル処理されたビットマップデータ223を生成するため、読まれたデータを復号化するステップと、を含む。次

に、符号化され、スクランブル処理されたデータ 224 を生成するため、チェッカーボード XOR 演算がそのデータに適用される。このデータは、次に、データ 225 を生成するためにスクランブル解除 227 され、その後、このデータは、元の生データ 226 を生成するため、リード・ソロモン復号化を受ける。或いは、スクランブル解除及び XOR 演算プロセスは、一緒に行ってもよく、別々のデータのパスを必要としない。上記のステップの各々は、以下で詳述される。図 37 を参照して説明したように、Art card インタフェースは、4 フェーズを有し、そのうちの最初の二つは、時間的に重大であり、画素データが CCD から読み出される間に行われなければならない：

フェーズ 1 Art card 上のデータエリア検出

フェーズ 2 CCD 画素に基づく Art card からのビットパターン検出と、バイトとしての書き込み

フェーズ 3 バイトパターンのスクランブル解除及び XOR 演算

フェーズ 4 データ復号化（リード・ソロモン復号）。

【0377】

これらの 4 フェーズを次に詳細に説明する。

【0378】

フェーズ 1：Art card 9 が CCD 34 を通過するとき、AI は、データエリアの左側で Arr card 上の特別ターゲットを確実に検出することにより、データエリアのスタートを検出しなければならない。特別ターゲットが検出され得なかった場合、カードは無効としてマークされる。この検出は、Art card 9 が CCD 34 を通過する間に、リアルタイムで行われる必要がある。

【0379】

必要に応じて、回転不変性が与えられる。この場合、ターゲットは、Art card の右側で繰り返されるが、上隅ではなく、下右隅に関連している。このようにして、もし、カードが「間違った」やり方で挿入された場合、ターゲットは、結局、正しい向きにされる。後述のフェーズ 3 は、データの向きを検出し、回転の可能性を考慮するように変更してもよい。

【0380】

フェーズ 2：データが一旦決定されると、主たる読み出しプロセスが始まり、CCD からの画素データを「Art card データ窓」に入れ、この窓からのビットを検出し、検出されたビットをバイトに組み立て、DRAM にバイトイメージを構築する。これは、Art card が CCD を通過する間に全て行われる必要がある。

【0381】

フェーズ 3：全ての画素が Art card データエリアから読み出されると、Art card モーター 37 は停止させられ、バイトイメージはスクランブル解除され、XOR 演算される。リアルタイム性能は要求されないが、このプロセスは、人間のオペレータを煩わせない程度に高速でなければならない。このプロセスは、2MB のスクランブル化されたビットイメージを取得し、スクランブル加除され／XOR 演算されたビットイメージを別々の 2MB イメージに書き込む。

【0382】

フェーズ 4：Art card 読み出しプロセスの最終フェーズは、リード・ソロモン復号化プロセスであり、2MB ビットイメージが 1MB の有効 Art card データエリアに復号化される。再び、リアルタイム性能は要求されないが、人間のオペレータからみて素早く復号化を行うことが必要である。復号プロセスが有効であるならば、カードは有効としてマークされる。復号に失敗した場合、ビットイメージ内のデータの複製の復号化が試みられ、プロセスは、成功するか、又はビットイメージ内にデータの複製イメージが存在しなくなるまで繰り返される。

【0383】

上記の 4 フェーズプロセスは、4.5MB の DRAM を必要とする。2MB はフェーズ 2 の出力のために確保され、0.5MB はフェーズ 1 及び 2 の間のスクラッチデータ用に確

10

20

30

40

50

保される。残りの 2 MB のスペースは、1 列当りに 4 7 2 5 バイトとして 4 4 0 列以上を保持することができる。實際上、読み出されている画素データはフェーズ 1 のアルゴリズムよりも数列分だけ先行しており、最悪のケースでは、約 1 8 0 列だけフェーズ 2 よりも遅れるが、4 4 0 列の限界の範囲内に十分に収まる。

【0384】

次に、各フェーズの実際の動作を詳細に説明する。

【0385】

フェーズ 1 - A r t c a r d 上のデータエリア検出

このフェーズは、A r t c a r d 9 上のデータエリアの左側を確実に検出することと関係がある。データエリアの正確な検出は、カードの左側に印刷された特殊ターゲットの正確な検出によって実現される。これらのターゲットは、特に、最大 1 度までの回転がある場合でも簡単に検出できるように設計されている。

【0386】

図 3 8 を参照すると、A r t c a r d 9 の左側の拡大図が示されている。このカードの左側は 1 6 個のバンド 2 3 9 に分割され、ターゲット、例えば、2 4 1 は各バンドの中心に設けられる。バンドは、バンドを分離するための線が引かれていないという点で論理的である。図 3 9 には、1 個のターゲットが示されている。ターゲット 2 4 1 は、単一の白色ドットを含む印刷された黒色正方形である。この考え方は、最初にできるだけ多数のターゲット 2 4 1 を検出し、次に、少なくとも 8 個の検出された白色ドットの場所を単一の論理的な直線に繋ぎ合わせるというものである。この考え方が実行できれば、データエリア 2 4 3 のスタートは、この論理的な直線から一定距離になる。実行できない場合、カードは無効であるとして拒否される。

【0387】

図 3 8 に示されるように、カード 9 の高さは 3 1 5 0 ドットである。ターゲット (Target 0) 2 4 1 は、データエリアの左上隅 2 4 4 から 2 4 ドットの一定距離だけ離れた場所に置かれるので、ターゲットは、1 9 2 ドット (5 7 6 画素) の等しいサイズの 1 6 個の領域のうちの最初の領域 2 3 9 の範囲に巧く収まり、カードの最後の画素領域にターゲットは存在しない。ターゲット 2 4 1 は、簡単に検出できるように十分に大きくしなければならないが、カードが 1 度回転したときに領域の外側へ出る程に大きくしてはならない。ターゲットの適当なサイズは、白色ドット 2 4 2 を含む 3 1 × 3 1 ドット (9 3 × 9 3 個の検出画素) の黒色正方形 2 4 1 である。

【0388】

最悪の 1 度の回転がある場合、1 列分のシフトは 5 7 画素毎に起こる。したがって、5 9 0 画素のサイズのバンドの場合、カードが最悪の回転状態であるならば、このシンボルの何れかの部分を、上部若しくは下部の 1 2 画素、又はバンドのその場所に配置することは不可能であり、或いは、シンボルは C C D 読み出し時に間違ったバンドで検出されるであろう。

【0389】

したがって、矩形の黒色部分が 5 7 画素の高さ (1 9 ドット) である場合、少なくとも 9 . 5 個の黒色画素は、C C D によって同一列で確実に読み出すことができる (画素の半分が 1 列に収まり、あとの半分が次の列に収まる最悪ケース)。同一列で少なくとも 1 0 個の黒色ドットが確実に読み出されるようにするため、高さを 2 0 ドットにする必要がある。黒色ドットのスタートのエッジでの誤検出に備えて、ドットの個数を 3 1 まで増加させ、ターゲットのローカル座標 (1 5 , 1 5) に置かれた白色ドットの両側に 1 5 ドットずつを置く。3 1 ドットは 9 1 画素であり、列内のシフトは最大でも 3 画素であり、5 7 6 画素のバンドの範囲内に容易に収まる。

【0390】

このように、各ターゲットは、3 1 × 3 1 ドット (9 3 × 9 3 画素) のブロックであり、各ターゲットは、
各列に 3 1 個の黒色ドットを含む 1 5 列 (9 3 画素のうちの 4 5 画素幅の列) と、

10

20

30

40

50

15個の黒色ドット（45画素）、1個の白色ドット（3画素）、及び更なる15個の黒色ドット（45画素）を含む1列と、
各列が31個の黒色ドットを含む15列（93画素のうちの45画素幅の列）と、
により構成される。

【0391】

ターゲット検出

ターゲットは、ドットを検出するのではなく、同時に1列ずつ画素の列を読み出すことにより検出される。所定のバンド内で、ターゲットの左側を構築する多数の隣接した黒色画素により構成された多数の列を探索することが必要である。次に、更なる黒色列の中心で白色領域を見つけることが期待され、最後に、ターゲットの中心の左側で黒色列が見つ

10

【0392】

画素の読み出しの際に優れたキャッシュ性能を得るためには8個のキャッシュラインが必要である。論理的な読み出し毎に、4回のサブ読み出しによって4個のキャッシュラインが満たされ、一方、その他の4個のキャッシュラインは使用される。これは、利用可能なDRAM帯域幅の13%までを効率的に使用する。

【0393】

図40に示されるように、ターゲットを検出する検出機構FIFOは、フィルタ245、ランレングスエンコーダ246、及びランダムアクセスのため最上部3要素（S1、S2及びS3）の特別な配線を必要とするFIFO247を含む。

20

【0394】

入力画素の列は、全てのターゲットが検出されるか、又は指定された数の列が処理されるまで、同時に1列ずつ処理される。列を処理するため、画素はDRAMから読み出され、0又は1を検出するためフィルタ245の中を通され、次に、ランレングス符号化246される。ビット値、及び同じ値の隣接したビットの個数は、FIFO247に入れられる。FIFO249の個々のエントリーは8ビットに収まり、7ビット250はランレングスを保持し、1ビット249は検出されたビットの値を保持する。

【0395】

ランレングスエンコーダ246は、576画素（192ドット）領域内の隣接した画素だけを符号化する。

30

【0396】

FIFO247の最上部3要素は、ランダムな順序でアクセス252され得る。これらのエントリーの（画素の）ランレングスは、以下の表に従って、ショート、ミディアム、及びロングの3個の値にフィルタ処理される。

【0397】

【表39】

ショート	白色ドット検出のため使用	ランレングス < 16
メディアム	ターゲットの中心で白色ドットの上下の黒色ランを検出するため使用	16 ≤ ランレングス < 48
ロング	ターゲットの中心ドットの左側及び右側の黒色のランレングスを検出するため使用	ランレングス ≥ 48

40

FIFO247の最上部の3個のエントリーに着目すると、3種類の興味深い特殊ケースが存在する。

【0398】

【表 4 0】

ケース 1	S1 = 白色ロング S2 = 黒色ロング S3 = 白色ミディアム／ロング	白色中心ドットの左側又は右側でターゲットの黒色列を検出
ケース 2	S1 = 白色ロング S2 = 黒色ミディアム S3 = 白色ショート 前の 8 行はケース 1	ケース 1 の一連の列を処理した場合、おそらくこの列に白色ドットが検出される。次のエントリーが黒色であることはわかっているが(又は、白色 S3 エントリーに含まれているが)、黒色画素の個数が問題である。次の FIFO の前進後にチェックによって検証する必要がある(ケース 3 を参照)。
ケース 3	Prev = ケース 2 S3 = 黒色ミディアム	白色ドットの一部が検出された。これらは約 3 であり、次に、ケース 1 のもう少しの列が予想される。

10

好ましくは、領域バンド毎に以下の情報が保持される。

【 0 3 9 9】

【表 4 1】

TaregetDetected	1 ビット
BlackDetectCount	4 ビット
WhiteDetectCount	3 ビット
PrevColumnStartPixel	15 ビット
TargetColumn ordinate	16 ビット(15 : 1)
TargetRow ordinate	16 ビット(15 : 1)
合 計	7 バイト(アドレッシングの簡単化のため 8 バイトに丸め)

20

全部で 7 バイトが与えられる。合計が 8 バイトであるならば、アドレス生成が簡単化される。そこで、16 個のエントリーは、 $16 * 8 = 128$ バイトが必要であり、これは、4 個のキャッシュラインとぴったり合う。アドレスレンジは、0.5 MB の DRAM エリアのスクラッチの内側に入るべきである。なぜならば、他のフェーズが残りの 4 MB データエリアを利用するからである。

30

【 0 4 0 0】

所定の画素列を処理し始めるとき、レジスタ値 S2StartPixel 254 は 0 にリセットされる。FIFO 内のエントリーが S2 から S1 へ進むとき、それらは、既存の S2StartPixel 値に加算され (255)、現在 S2 で定義されているランの正確な位置を与える。FIFO 内の 3 個の興味深いケースの各々を調べると、S2StartPixel はターゲットの黒色エリアのスタートを判定するため使用可能であり (ケース 1 及び 2)、ターゲットの中心にある白色ドットのスタートを判定するため使用可能である (ケース 3)。列を処理するアルゴリズムは以下のように表される。

40

【 0 4 0 1】

【表 4 2】

1	TargetDetected [0 - 15] := 0 BlackDetectCount [0 - 15] := 0 WhiteDetectCount [0 - 15] := 0 TargetRow [0 - 15] := 0 TargetColumn [0 - 15] := 0 PrevColStartPixel [0 - 15] := 0 CurrentColumn := 0
2	プロセス Column を実行
3	CurrentColumn++
4	If (CurrentColumn <= LastValidColumn) Goto 2

10

列の処理に關与するステップ（プロセスColumn）は以下の通りである。

【0 4 0 2】

【表 4 3】

1	S2StartPixel := 0 FIFO := 0 BlackDetectCount := 0 WhiteDetectCount := 0 ThisColumnDetected := FALSE PrevCaseWasCase2 := FALSE
2	If (! TargetDetected [Target]) & (! ColumnDetected [Target]) ProcessCases EndIf
3	PrevCaseWasCase2 := Case = 2
4	FIFOを進める

20

30

3 種類のケース（処理ケース）の各々に対する処理は以下の通りである。

【0 4 0 3】

ケース 1 :

【0 4 0 4】

【表 4 4】

BlackDetectCount [target] < 8 OR WhiteDetectCount [Target] = 0	$\Delta := \text{ABS} (\text{S2StartPixel} - \text{PrevColStartPixel} [\text{target}])$ If (0 <= Δ <= 2) BlackDetectCount [Target] ++ (最大値 = 8) Else BlackDetectCount [Target] := 1 WhiteDetectCount [target] := 0 EndIf PrevColStartPixel [Target] := S2StartPixel ColumnDetected [Target] := TRUE BitDetected = 1
BlackDetectCount [target] >= 8 WhiteDetectCount [Target] != 0	PrevColStartPixel [Target] := S2StartPixel ColumnDetected [Target] := TRUE BitDetected = 1 TargetDetected [Target] := TRUE TargetDetected [Target] := CurrentColumn - 8 - (WhiteDetectCount [Target] /2)

10

20

ケース 2 :

ケース 3 を識別するため PrevCaseWasCase2 フラグ (前のケースはケース 2 であった) をセットする以外に特別な処理は記録されない (上記の列を処理するステップ 3 を参照)。

【0 4 0 5】

ケース 3 :

【0 4 0 6】

【表 4 5】

PrevCaseWasCase2 = TRUE BlackDetectCount [Target] >= 8 WhiteDetectCount = 1	If (WhiteDetectCount [Target] < 2) TargetRow [Target] = S2StartPixel + (S2RunLength/2) EndIf $\Delta := \text{ABS} (\text{S2StartPixel} - \text{PrevColStartPixel} [\text{Target}])$ If (0 <= Δ < 2) WhiteDetectCount [Target] ++ Else WhiteDetectCount [Target] := 1 EndIf PrevColStartPixel [Target] := S2StartPixel ThisColumnDetected := TRUE BitDetected = 0

30

40

所与の列を処理する最後に、現在の列がターゲット検出のための列の最大番号と比較される。許容された列の番号を超えたとき、検出されたターゲットの個数をチェックする必要がある。検出されたものが 8 個未満であるならば、カードは無効であると見なされる。

【0 4 0 7】

プロセス ターゲット

ターゲットが検出された後、ターゲットを処理する必要がある。全部のターゲットが利用可能であるか、又は一部のターゲットだけが利用可能である。一部のターゲットは誤って検出された可能性もある。

【0 4 0 8】

この処理のフェーズは、できるだけ多数のターゲットの中心を通る数学的な直線を決定す

50

る。その直線が通過するターゲットの個数が多いほど、より高い信頼度でターゲット位置が検出される。限界は8ターゲットに設定される。直線が少なくとも8個のターゲットを通過する場合、正しい直線である見なされる。

【0409】

力づくではあるが、簡単なアプローチを採用することは差し支えない。なぜならば、そのように行うための時間があり（以下を参照）、複雑さを低減することによりテストが簡単化されるからである。Target0とTarget1（両方のターゲットが有効であると見なされる場合）の間で直線を決定し、この直線上にあるターゲットの個数を決定する必要がある。次に、Target0とターゲット2の間の直線を決定し、このプロセスを繰り返す。最終的に、Target1と2の間の直線、1と3の間の直線等と同じプロセスを実行し、以下同様に続き、最終的に、Target14とTarget15の間の直線に同じプロセスを実行する。全てのターゲットが検出された場合を想定すると、 $15 + 14 + 13 + \dots = 90$ 回の計算（各計算の組は16回のテストを必要とするので、実際には、1440回の計算）を実行し、その直線に沿って最大個数のターゲットが検出された直線を選択する必要がある。ターゲット位置を見つけるためのアルゴリズムは、例えば、以下のようになる：

```

TargetA := 0
MaxFound := 0
BestLine := 0
While (TargetA < 15)
    If (TargetA is Valid)
        TargetB:= TargetA + 1
    10
    While (TargetB<= 15)
    If (TargetB is valid)
    CurrentLine := line between TargetA and TargetB
    TargetC := 0;
    While (TargetC <= 15)
        If (TargetC valid AND TargetC on line AB)
            TargetsHit++
            20
        EndIf
        If (TargetsHit > MaxFound)
            MaxFound := TargetsHit
            BestLine := CurrentLine
        EndIf
        TargetC++
    EndWhile
    30
    EndIf
    TargetB ++
    EndWhile
    EndIf
    TargetA++
EndWhile
    40

If (MaxFound < 8)
    Card is Invalid

Else
    Store expected centroids for rows based on BestLine

EndIf

```

entLine 2 6 0を決定するため、ターゲット 2 6 1、2 6 2と、ターゲット A の位置との間で Δrow (2 6 4) 及び Δcolumn (2 6 3) を計算することが必要である。 Δrow 及び Δcolumn を加算することにより、Target0からTarget1、以下同様、へ移動することが可能になる。検出された（実際に検出されたならば）TargetNの位置は、直線上のTargetNの計算された予測位置と比較することができ、許容範囲内に収まるならば、TargetNは直線上にあることが判定される。

【0 4 1 0】

Δrow 及び Δcolumn は、

$$\Delta \text{row} = (\text{row}_{\text{TargetA}} - \text{row}_{\text{TargetB}}) / (B - A)$$

$$\Delta \text{column} = (\text{column}_{\text{TargetA}} - \text{column}_{\text{TargetB}}) / (B - A)$$

により計算される。次に、Target0の位置を、

$$\text{row} = \text{row}_{\text{TargetA}} - (A * \Delta \text{row})$$

$$\text{column} = \text{column}_{\text{TargetA}} - (A * \Delta \text{column})$$

により計算する。

【0 4 1 1】

また、(row,

column)を実際の $\text{row}_{\text{Target0}}$ 及び $\text{column}_{\text{Target0}}$ と比較する。ある予測ターゲットから次のターゲット（例えば、Target0からTarget1）へ移動するためには、 Δrow 及び Δcolumn をそれぞれrow及びcolumnに加算すればよい。各ターゲットが直線上にあるかどうかをチェックするため、Target0の予測位置を計算し、次に、各ターゲットの座標に対して1回の加算及び1回の比較を実行することが必要である。

【0 4 1 2】

1 6 個全てのターゲットと最大で90本の直線との比較の終了後、得られる結果は有効ターゲットを通る最良直線である。その直線が少なくとも8個のターゲットを通過するならば（即ち、 $\text{MaxFound} \geq 8$ ）、直線を形成するために十分なターゲットが検出されたことが認められるので、カードを処理することができる。最良直線の通過するターゲットが8個未満であるならば、カードは無効であると見なされる。

【0 4 1 3】

結果として生じるアルゴリズムは、 Δrow 及び Δcolumn を計算するための180回の除算と、Target0の位置を計算するための180回の乗算／加算と、2880回の加算／比較と、を要する。この処理を実行するための時間は、36列の画素データを読み出すために要する時間、即ち、3374892 nsである。加算は除算よりも所要時間が短いという事実までを考慮しない場合、3240回の数学的演算を3374892 ns内に実行しなければならない。これは、1演算あたりに約1040 ns、即ち、104サイクルに相当する。したがって、CPUは、ターゲットの全ての処理を無事に実行することが可能であり、設計の複雑さは低減される。

【0 4 1 4】

データエッジ境界及びクロックマークに基づく重心の更新

ステップ0：データエリアの位置検出

Target0（図38の241）からデータエリアの左上境界244までの行及び列に関する距離は所定の固定距離であり、更に、垂直クロックマーク276までには1ドットの列がある。したがって、前の段階で見つかったTargetA、 Δrow 及び Δcolumn （ Δrow 及び Δcolumn はターゲット間の距離を表す）は、上述のようにTarget0の重心又は予測位置を計算するため使用される。

【0 4 1 5】

Target0からデータエリアまでの固定画素オフセットはターゲット間の距離（ターゲット間は192ドット、Target0とデータエリア243の間は24ドット）と関連付けられるので、単に、 $\Delta \text{row}/8$ をTarget0の重心の列座標に加算すればよい（ドットのアスペクト比

は 1 : 1) 。最上部座標は、

$$(\text{column}_{\text{DotColumnTop}} = \text{column}_{\text{Target0}} + (\Delta \text{row}/8)$$

$$(\text{row}_{\text{DotColumnTop}} = \text{row}_{\text{Target0}} + (\Delta \text{column}/8)$$

 として定義される。

【0416】

次の Δrow 及び Δcolumn は、ターゲット間のドット数によってそれらを除算することにより、（ターゲット間ではなく）単一系列内のドット間の画素数を与えるために更新される：

$$\Delta \text{row} = \Delta \text{row}/192$$

$$\Delta \text{column} = \Delta \text{column}/192$$

10

更に、currentColumnレジスタ（フェーズ2を参照）には-1をセットすることにより、ステップ2の後、フェーズ2が開始したとき、currentColumnレジスタは-1から0へインクリメントする。

【0417】

ステップ1：初期重心デルタ (Δ) 及びビット履歴の書き出し
 これは、フェーズ2のために必要なセットアップ情報を書き込むだけである。

【0418】

これは、各行のすべての Δrow 及び Δcolumn のエントリーと、ビット履歴に0を書き込むことによって実現できる。ビット履歴は実際には予測ビット履歴である。なぜならば、クロックマーク列276の左側には境界列277があり、その前には、白色エリアが存在することがわかっているからである。したがって、ビット履歴は、011、010、011、010等である。

20

【0419】

ステップ2：読み出された実際の画素に基づく重心の更新
 ビット履歴は、予測クロックマーク及びデータ境界に従ってステップ1でセットアップされる。各ドット行の実際の重心は、予測データを実際の画素値と比較することによって、より正確にセットされる（初期的には0である）。重心更新機構は、フェーズ2のステップ3を実行するだけで実現される。

【0420】

フェーズ2ー読み出された画素に基づく Art card からのビットパターン検出及びバイトとしての書き込み
 Art card 9からのドットは、3列に亘って検出された少なくとも9個の画素が提示されることを要求するので、検出された画素列毎にドット検出計算を実行するポイントは殆ど無い。処理に要する時間は、平均ドット出現に亘って平均化し、利用可能な処理時間を最大化した方がよい。これにより、Art card 9からのドットの列の処理を、Art card から3列のデータを読み出すために要する時間内に収めることが可能になる。最も可能性の高いケースでは、ドットを提示するために4列を要するが、4番目の列は、あるドットの最後の列であり、且つ次のドットの最初の列である。したがって、処理は3列だけに制限されるべきである。

30

40

【0421】

CCDからの画素は、利用可能な時間の13%内でDRAMに書き込まれるので、83%の時間、即ち、 93747×3 の83% = 281241 ns の83% = 233430 ns は、ドットの1列を処理するために利用可能である。

【0422】

利用可能な時間内に、3150ドットを検出し、それらのビット値をメモリの生データエリアに書き込むことが必要である。したがって、この処理は以下のステップを必要とする：

Art card 上のドットの列毎に、
 ステップ0：次のドット列へ進む；

50

ステップ1: Art card ドット列の上端及び下端を検出する (クロックマークをチェックする) ;

ステップ2: ドット列を処理し、ビットを検出し、ビットを適切に記憶する ;

ステップ3: 重心を更新する。

【0423】

Art card の論理的なドット列を処理し、論理的なドット列は165画素に亘ってシフトする可能性があるので、最悪のケースでは、少なくとも165列がDRAMに読み込まれるまで、最初の列を処理できない。したがって、フェーズ2は、読み出しプロセスが終了した後、同じ時間で終了する。最悪ケースの時間は、 $165 * 93747 \text{ ns} = 15468255 \text{ ns}$ 、即ち、0.015秒である。

10

【0424】

ステップ0: 次のドットへ前進

次のドットの列へ進むため、 Δrow 及び Δcolumn を dotColumnTop に加算し、列の上端のドットの重心を得る。最初にこれを行うとき、ビットイメージ画像データエリアの左側のクロックマーク列276の所にいるので、第1のデータ列へ進む。 Δrow 及び Δcolumn は、1列内のドット間の距離を表すので、ドット列の間を移動するために、 Δrow を $\text{column}_{\text{dotColumnTop}}$ に加算し、 Δcolumn を $\text{row}_{\text{dotColumnTop}}$ に加算することが必要である。

【0425】

処理中の列番号を追跡するため、列番号は、CurrentColumnという名称のレジスタに記録される。センサが次のドット列へ進むたびに、CurrentColumnレジスタをインクリメントする必要がある。CurrentColumnレジスタが最初にインクリメントされるとき、それは、-1から0へインクリメントされる (フェーズ1のステップ0を参照)。CurrentColumnレジスタは、(maxColumnsに達したときに) 読み出しプロセスを終了するときを決定し、(8ドット列毎に1回ずつ) 8ビット全部がバイトに書き込まれた後、DataOutPointerを次のバイト情報の列へ進めるために使用される。下位3ビットは、現在のバイトにどのビットまで入るかを決定する。

20

【0426】

ステップ1: Art card ドット列の上端及び下端の検出

Art card からのドット列を処理するため、列の上端及び下端を検出することが必要である。列は、(局所的な歪み等を除いて) 列の上端と下端の間に直線を形成すべきである。初期的に、dotColumnTopはクロックマーク列276を指定する。予測値をトグルさせ、予測値をビット履歴に書き込み、ステップ2へ進むだけでよく、その最初のタスクは、列の最初のデータドットに到達するため、 Δrow 及び Δcolumn の値を dotColumnTop に加算することである。

30

【0427】

ステップ2: Art card のドット列を処理

列の上端及び下端の重心が画素座標に与えられたならば、列は、それらの間に歪み等を原因とする変動ができる限り少ない直線を形成する。

【0428】

処理が列の上端 (上端重心座標) でスタートし、列の下端へ下がる場合を想定すると、後続の予測ドット重心は、

40

$$\text{row}_{\text{next}} = \text{row} + \Delta \text{row}$$

$$\text{column}_{\text{next}} = \text{column} + \Delta \text{column}$$

として与えられる。

【0429】

これは、列の次のドットの予測重心のアドレスを与える。しかし、局所的な歪み及び誤差を考慮するため、直前に所定の行で検出したドットに基づく別の Δrow 及び Δcolumn を加算する。このようにして、列の上端から下端を繋ぎ合わせる直線からあるパーセンテージの最大のドリフトに累積する小さいドリフトを考慮することができる。

【0430】

50

したがって、行毎に2個の値を保持するが、行履歴はこのフェーズのステップ3で使用されるので、それらの2個の値は別々のテーブルに格納する：

* Δrow 及び Δcolumn （それぞれ2 @ 4ビット＝1バイト）；

* 行履歴（1行毎に3ビット、1バイト当たり2行が格納される）。

【0431】

行毎に、重心への変化を決定するため、 Δrow 及び Δcolumn を読み出すことが必要である。読み出しプロセスは、帯域幅の5%と2個のキャッシュラインを要する：

$76 * (3150 / 32) + 2 * 3150 = 13824 \text{ ns} = \text{帯域幅の5\%}$ 。

【0432】

重心が決定されると、重心の周辺画素は、ドットの状態、即ち、ビットの値を検出するために調べられる。換言すると、ドットは、 4×4 画素のエリアを覆う。しかし、ドットの解像度の3倍の解像度でサンプリングしているので、ドットの状態、即ち、ビットの値を検出するために必要な画素数はこれよりも遙かに少ない。1回につきアクセスすべき画素列は、3列に過ぎない。

【0433】

1%の回転による画素ドリフトのある最悪ケースでは、重心は57画素行毎に1列ずつシフトするが、ドットの直径が3ドットであるため、ある列は、171画素行（ $3 * 57$ ）に対して有効であろう。1バイトは2画素を含むので、各バッファ読み出し（4個のキャッシュライン）で有効なバイト数は、最悪の場合に（128の読み出しのうちの）86であろう。

【0434】

ビットが検出されると、ビットはDRAMへ書き出すことが必要である。DRAM遅延を最小限に抑えるため、8列からのビットが隣接したバイトのセットとして格納される。所与のドット列からの全てのビットは、データバイト内の次のビット位置に対応するので、そのバイトの古い値を読み出し、新しいビット内でシフトしOR演算し、そのバイトを書き戻すことが可能である。

【0435】

読み出し／シフト及びOR演算／書き込みプロセスは2個のキャッシュラインを必要とする。

【0436】

所定の行に対するビット履歴を更新するとき、そのビット履歴を読み出し、書き込むことが必要である。1行当たりに必要なビット履歴は3ビットだけであり、2行の履歴を単一バイトに格納することが可能である。読み出し／シフト及びOR演算／書き込みプロセスは2個のキャッシュラインを必要とする。

【0437】

ビット検出及び蓄積のために要する総帯域幅は以下の表に掲載されている。

【0438】

【表46】

重心 Δ 読み出し	5%
3列の画素データを読み出し	19%
検出されたビットをバイトバッファに読み出し／書き込み	10%
ビット履歴を読み出し／書き込み	5%
合 計	39%

ドットの検出

重心を与えるドットの値（即ち、ビットの値）を検出するプロセスは、3個の画素値を検査し、ルックアップテーブルから結果を取得することによって実現される。この処理は、

かなり簡単であり、図 4 2 に示されている。ドット 2 9 0 の半径は約 1.5 画素である。したがって、重心を支える画素 2 9 1 は、その画素内の重心の実際の位置とは無関係に、完全にそのドット値であろう。重心が画素 2 9 1 の中心と正確に一致するならば、重心画素の上画素 2 9 2 及び下画素 2 9 3、並びに、重心画素の左画素 2 9 4 及び右画素 2 9 5 は、大部分そのドット値であろう。重心が画素 2 9 5 の正確な中心から離れているならば、中心画素以外の画素がそのドットによって 100% 覆われる可能性が高い。

【0439】

図 4 2 には、中心の左及び下へ偏っている重心だけが示されているが、同じ関係が中心の上側及び右側の重心に対しても成り立つことは明白である。ケース 1 では、重心は、中央画素 2 9 5 の中心と正確に一致する。中心画素 2 9 5 はそのドットによって完全に覆われ、上、下、左、及び右の画素もそのドットによって十分に覆われている。ケース 2 では、重心は、中央画素 2 9 1 の中心の左側にある。中心画素は、依然としてドットによって完全に覆われ、中心の左の画素 2 9 4 はドットによって完全に覆われている。上画素 2 9 2 及び下画素 2 9 3 は依然として十分に覆われている。ケース 3 では、重心は、中央画素 2 9 1 の中心の下側にある。中心画素 2 9 1 は、依然としてドット 2 9 1 によって完全に覆われ、中心の左の画素はドットによって完全に覆われている。中心の左画素 2 9 4 及び右画素 2 9 5 は依然として十分に覆われている。ケース 4 では、重心は中央画素の中心の左下にある。中心画素 2 9 1 は依然としてドット 2 9 1 によって完全に覆われ、中心の左画素 2 9 4、及び中心の下画素 2 9 4 はドットによって完全に覆われている。

【0440】

重心を更新するアルゴリズムは、以下の 3 個の代表的な画素を選択し、ドットの値を決定するため、中央画素 2 9 1 の中心から重心までの距離を使用する：

第 1 画素：重心を含む画素

第 2 画素：重心の X 座標（列値）が $< 1/2$ である場合、第 1 画素の左側の画素であり、さもなければ、第 1 画素の右側の画素

第 3 画素：重心の Y 座標（行値）が $< 1/2$ である場合、第 1 画素の上側の画素であり、さもなければ、第 1 画素の下側の画素。

【0441】

図 4 3 に示されるように、各画素の値は、予め計算されたルックアップテーブル 3 0 1 へ出力される。3 個の画素は 12 ビットのルックアップテーブルへ供給され、このルックアップテーブルはドットの値、即ち、オン又はオフを示す単一ビットを出力する。ルックアップテーブル 3 0 1 は、チップ定義時に構築され、約 500 ゲートにまとめられる。ルックアップテーブルは、簡単な閾値テーブルでもよいが、但し、中心画素（第 1 画素）は重く重み付けされる。

【0442】

ステップ 3：列において行毎に重心 Δs の更新

Δs 処理の考え方は、現在列において、行毎に予測重心位置で完全なドットを生成するため、前のビット履歴を使用することである。実際の（CCDからの）画素は、予測完全画素と比較される。両方が一致する場合、実際の重心位置は正確に予測位置に含まれるので、重心 Δs は有効であり、更新する必要がない。さもなければ、予測重心位置を実際のデータに最も良く適合させるため、重心 Δs を更新するプロセスを行うことが必要である。新しい重心 Δs は次の列のドットを処理するために使用される。

【0443】

重心 Δs の更新は、下記の理由で、ステップ 2 からの後続のプロセスとして実行される：なぜならば、

設計の複雑さを軽減し、それにより、十分な帯域幅が残っているフェーズ 1 のステップ 2 として実行でき、DRAMバッファを再使用できるようになり、

重心更新に必要な全てのデータが特殊なパイプライン化を行うことなく、プロセスのスタート時に利用できることが保証されるからである。

【0444】

重心 Δ は、複雑さを低減するため、 Δ column及び Δ rowとして処理される。

【0445】

所与のドットは直径が3ドットであるが、 4×4 の画素エリアに出現する可能性が高い。しかし、あるドットのエッジは、結果的に、次のドットのエッジと同じ画素に含まれる。このため、重心更新のためには、所与の単一ドットに関する情報以外の情報が必要である。

【0446】

図44には、前の列からの単一のドット310が所与の重心311と共に示されている。本例では、ドット310は、4個の画素列312から315の範囲に Δ が広がり、実際上、前のドット列のドット（座標=(Prevcolumn, CurrentRow)）の一部は、現在行のドットの現在列に入り込んでいる。現在の行及び列のドットが白色であるならば、前の列のドットからのドット情報（現在の列のドットは白色）しか存在しないので、前のドット列から最も右側にある画素列314は小さい値であると予測される。このことから、この画素列315における画素値が高くなればなるほど、重心がより右側になることがわかる。勿論、右側のドットも黒色であるならば、サブピクセルから情報を獲得できないので、重心を調整することができる。同様のことは、ドット座標(Prevcolumn, CurrentRow)における左、上及び下のドットについても成り立つ。

【0447】

このことから、最大で5個の画素列及び画素行が必要であることがわかる。行及び列の重心 Δ sを別々に考慮し、それらを90度だけ回転した同じ問題として取り扱うことにより、状況を簡単化することが可能である。

【0448】

水平方向のケースを最初に説明すると、予測画素が検出画素と一致しない場合、列重心 Δ sを変更することが必要である。ビット履歴から、現在ドット列の現在行に対して見つかったビットの値、直前のドット列、及び二つ前のドット列がわかる。予測重心位置もわかっている。これらの二組の情報を使用することにより、読み出しが完全であるならば、20ビットの予測ビットパターンを生成することが可能である。20ビットのビットパターンは水平方向次元で5画素毎の予測 Δ 値を表現する。1番目のニブルは、最も左側のドットの最も右側の画素を表現する。次の3個のニブルは、前の列からドット310の中心を通る3画素を表現し、最後のニブルは（現在列から）最も右側のドットの最も左側の画素317を表現する。

【0449】

予測重心が画素の中心である場合、20ビットパターンは以下の表に基づくことが予測される。

【0450】

10

20

30

【表 4 7】

ビット履歴	予測画素
000	00000
001	0000D
010	0DFD0
011	0DFDD
100	D0000
101	D000D
110	DDFD0
111	DDFDD

10

20

30

40

50

中心ドットの左側及び右側の画素は、ビットが0であるか、又は1であるかに応じて、それぞれ、0又はDである。中心の3画素は、ビットが0であるか、又は1であるかに応じて、それぞれ、000又はDFDである。これらの値は、所与の画素に対してドットによって占められた物理的なエリアに基づいている。画素の正確な中心から重心までの距離に依存して、僅かにシフトしたデータを予測し、これは、実際に中心画素の両側の画素だけに影響を与える。16通りの可能性があるので、中心からの距離を16で除算し、予測画素をシフトさせるためその量を使用することができる。

【0451】

20ビットの5個の画素予測値が決定されると、それは読み出された実際の画素と比較される。これは、画素単位で、読み出された実際の画素から予測画素を減算し、最後に、予測Δ値からの距離を獲得するため、それらの差をまとめて加算することにより行われる。

【0452】

図45は、上記アルゴリズムを実現する一形態の説明図であり、ビット履歴322及び重心小数コンポーネント323を受け取り、対応した20ビットの数324を出力するルックアップテーブル320が含まれ、20ビットの数324は、中心画素入力326から減算321され、画素差327が生成される。

【0453】

このプロセスは、予測重心に関して、Δcolumnの量1による重心の左シフト及び右シフトに対して1回ずつ実行される。実際の画素からの差が最も小さい重心は、「勝者」と見なされ、それに応じてΔcolumnが更新される（「変更無し」が望ましい）。その結果として、Δcolumnは、ドット列毎に1よりも大きく変更されることはない。

【0454】

このプロセスは垂直方向画素に対しても繰り返され、その結果としてΔrowが更新される。

【0455】

ここで、並列化できる見通しはかなり高い。ACPユニット31に対して選択されたクロックのレートに依存して、これらのユニットは直列に配置することができ（これにより、3種類のΔのテストは連続したクロックサイクルで行われる）、又は並列に配置することができ、これにより、3種類のテストを同時に行うことができる。クロックレートが十分に高速であるならば、並列化の必要性はない。

【0456】

帯域幅初期化

Δsの古いΔを読み出し、それらを再度書き出すことが必要である。これは、帯域幅の10%を要する：

$2 * (76 * (3150 / 32) + 2 * 3150) = 27648 \text{ ns} = \text{帯域幅の } 10\%。$

【0457】

Δs を更新するとき、所与の列のビット履歴を読み出すことが必要である。各バイトは2個の行のビット履歴を含むので、帯域幅の2.5%を要する：

$76 * ((3150 / 32) / 2) * 32 + 2 * (3150 / 2) = 4085 \text{ ns} = \text{帯域幅の } 2.5\%。$

【0458】

1%の回転によって画素がドリフトする最悪ケースでは、重心は57画素行毎に1列だけシフトするが、ドットの直径は3画素であるため、所与の画素列は、181画素行(3*57)に対して有効である。バイトは2画素を収容するので、キャッシュ読み出しにおいて有効なバイト数は最悪のケースで(128回の読み出しのうちの)86である。5列に対する最悪ケースのタイミングは、したがって、31%帯域幅である：

$5 * ((9450 / (128 * 2)) * 320) * 128 / 86 = 88112 \text{ ns} = \text{帯域幅の } 31\%。$

【0459】

重心 Δ を更新するために必要な総帯域幅は以下の表に掲載されている。

【0460】

【表48】

読み出し／書き込み重心 Δ	10%
読み出しビット履歴	2.5%
5列の画素データの読み出し	31%
合 計	43.5%

フェーズ2のメモリ使用量

2MBのビットイメージDRAMエリアは、フェーズ2処理中に読み書きされる。2MB画素データDRAMエリアは読まれる。

【0461】

0.5MBのスクラッチDRAMエリアは行データを格納するため使用される。即ち、

【0462】

【表49】

重心配列	24ビット (16:8)*2*3150 = 18900 バイト
ビット履歴配列	3ビット*3150 エントリー (1バイト当たり2エントリー) = 1575 バイト

フェーズ3ー生データのスクランブル解除及びXOR演算

図37を参照すると、復号化の次のステップは、生データのスクランブル解除及びXOR演算である。Art cardから取得されるような2MBバイト画像は、スクランブル処理され、XOR演算された形式である。これは、フェーズ4におけるリード・ソロモンデコーダに必要なビットイメージを得るためにスクランブル解除及び再XOR演算を必要とする。

【0463】

図46を参照すると、スクランブル解除プロセス330は、2MBのスクランブル付きのバイト画像331を取得し、スクランブル解除された2MB画像332を書き込む。このプロセスは、同じ場所で合理的に実行することが不可能であるため、2組の2MBエリアが利用される。スクランブル付きのデータ331は、16×16配列に並べられたシンボ

ルブロック順であり、シンボルブロック 0 (3 3 4) は、ランダムな順序の、あらゆる符号語からの全てのシンボル 0 を含む。シンボルブロック 1 は、ランダムな順序で全ての符号語からの全てのシンボル 1 を含み、以下同様である。2 5 5 個のシンボルしかないので、2 5 6 番目のシンボルブロックは、現時点では使用されない。

【0 4 6 4】

線形フィードバックシフトレジスタは、シンボルブロック、例えば、シンボルブロック 3 3 4 内の位置と、そのシンボルブロックが由来する符号語、例えば、符号語 3 5 5 との間の関係を決するため使用される。これは、元の Art card 画像を生成したときと同じシードが使用される限り機能する。時間のボトルネックは、DRAM の非順次アドレスへの読み出し／書き込みの準備完了を待機することであるため、他のソースラインからのバイトと、0 x A A 及び 0 x 5 5 との X O R 演算は、(時間的に) 実質的に制約がない。

10

【0 4 6 5】

スクランブル解除及び X O R 演算プロセスのタイミングは、実質的に、2 M B のランダムバイト読み出しと、2 M B のランダムバイト書き込みであり、即ち、 $2 * (2 M B * 7 6 n s + 2 M B * 2 n s) = 3 2 7 1 5 5 7 1 2 n s$ 、つまり、約 0 . 3 3 秒である。このタイミングはキャッシュ処理を想定していない。

【0 4 6 6】

フェーズ 4 - リード・ソロモン復号

このフェーズはループであり、ビットイメージ内のデータのコピーに対して繰り返され、復号が成功するか、又は復号しようとするコピーが無くなるまで、データのコピーをリード・ソロモン復号モジュールへ送る。

20

【0 4 6 7】

使用されるリード・ソロモンデコーダは、適切にプログラミングされた V L I W プロセッサでもよく、或いは、L S I ロジックの L64712 のような別個のハードワイヤードでもよい。L64712 は、1 秒当たり 5 0 M ビット (毎秒約 6 . 2 5 M B) のスループットを備えているので、時間は、2 M B の読み出し及び 1 M B の書き込みのためのメモリアクセス時間 (順次アクセスの場合 5 0 0 M B / 秒) ではなく、むしろ、リード・ソロモンデコーダの速度によって制約される。最悪ケースで要する時間は、2 / 6 . 2 5 秒 = 約 0 . 3 2 秒である。

【0 4 6 8】

フェーズ 5 V a r k スクリプトの実行

A r t c a r d 9 を読み出し、それを復号化するために要する総時間は、約 2 . 1 5 秒である。ユーザにとっての明らかな遅れは、実際には、0 . 6 5 秒 (フェーズ 3 及び 4 の合計) に過ぎない。なぜならば、A r t c a r d は、1 . 5 秒後に動きを止めるからである。

30

【0 4 6 9】

A r t c a r d がロードされると、A r t v a r k スクリプトは解釈される。直ちにスクリプトを実行するのではなく、スクリプトは、「プリント」ボタン 1 3 (図 1) を押したときに限り実行される。スクリプトを実行するために取られる措置は、スクリプトの複雑さに大きく依存し、プリントボタン及び実際のプリントボタンを押下し、実際のプリントが行われるまでの間に認識される遅れを考慮しなければならない。

40

【0 4 7 0】

代替 A r t c a r d フォーマット

勿論、他の a r t c a r d のフォーマットも考えられる。以下では、多数の好適な特徴を備えたこのような新しい代替 a r t c a r d フォーマットについて説明する。以下では、代替 A r t c a r d データフォーマット、ユーザデータを代替 A r t c a r d 上のドットにマッピングする機構、及びリソースが不足している組込型システムで使用するための高速代替 A r t c a r d 読み出しアルゴリズムを説明する。

【0 4 7 1】

代替 A r t c a r d の概要

50

代替 A r t c a r d は、組込型アプリケーション及び P C 型アプリケーションの両方で使用され、大容量のデータ又はコンフィギュレーション情報への使い易いインタフェースを提供する。

【0472】

代替 A r t c a r d の裏面は（データを保存するので）アプリケーションとは無関係に同一の外観を有するが、代替 A r t c a r d の表面はアプリケーションに依存している。表面はアプリケーションとの関係においてユーザに意味がなければならない。

【0473】

代替 A r t c a r d 技術は、印刷解像度とは独立にすることができる。カード上にドットとしてデータを保存する考え方は、（解像度を上げることによって）同じ空間により多くのドットを配置することができるならば、それらのドットがより多くのデータを表現し得るということを意味するに過ぎない。好適な実施例は、例示的な A r t c a r d としての 86mm×55mm カード上で 1600dpi の印刷を利用することを想定するが、他の等価的なレイアウト、他のカードサイズのためのデータサイズ、及び／又は、他の印刷解像度に決定することは容易である。印刷解像度とは無関係に、読み出し技術はそのまま維持される。全ての復号化及びその他のオーバーヘッドを考慮した後、代替 A r t c a r d は、1600dpi までの印刷解像度で、最大で 1 メガバイトのデータを保存することが可能である。代替 A r t c a r d は、1600dpi を超える印刷解像度で、数メガバイトのデータを保存することが可能である。以下の二つの表は、ある印刷解像度に対する代替 A r t c a r d の実効データ記憶容量を要約するものである。

【0474】

代替 A r t c a r d のフォーマット

代替 A r t c a r d 上のデータ構造は、データの復元を補助するため特別に設計されている。このセクションは代替 A r t c a r d のデータ（裏）面のフォーマットを説明する。

【0475】

ドット

代替 A r t c a r d のデータ面のドットはモノクロでもよい。例えば、黒色ドットが、所定の望ましい印刷解像度で白色背景に印刷される。その結果として、「黒色ドット」は物理的に「白色ドット」から区別される。図 47 には、黒色ドット及び白色ドットの拡大図の様々な例が示されている。白色背景上の黒色ドットのモノクロ方式は、好ましくは、ブラー効果のある読み出し環境においてダイナミックレンジを最大にするために選択される。黒色ドットが特定のピッチ（例えば、1600dpi）で印刷されているとしても、ドット自体は、ドットが隣接させて印刷されたときに、隣接した直線が作成されるように、少し大きくされる。ドットは、実際には滲みの結果として一つに併合されるかもしれないが、図 47 の例示的な画像では併合されていない。黒色のキザキザはもっと取り除かれるであろう。好適な実施例に記述された代替 A r t c a r d は、ドットサイズの柔軟に変化することを許容するが、最良の結果を得るためには、正確なドットサイズ、及び特定の印刷技術用のインク／印刷動作をより詳細に調べるべきである。

【0476】

この A r t c a r d の実施例の説明中、用語ドットは、代替 A r t c a r d 上に物理的に印刷されたドット（インク、熱、電子写真、ハロゲン化銀等）を意味する。代替 A r t c a r d リーダーが代替 A r t c a r d を走査するとき、ドットは、ナイキストの定理を満たすために、少なくとも印刷された解像度の 2 倍でサンプリングされる必要がある。用語画素は、代替 A r t c a r d リーダー装置からのサンプル値を意味する。例えば、1600dpi のドットが 4800dpi で走査されるとき、ドットの各次元に 3 個の画素、即ち、1 ドット当たり 9 個の画素が存在する。サンプリングプロセスは後述される。

【0477】

図 48 を参照すると、典型的な代替 A r t c a r d のデータ面 1101 が示されている。各代替 A r t c a r d は、白色境界領域 1103 によって囲まれた「アクティブ」領域 1102 により構成される。白色境界 1103 は、データ情報を格納しないが、代替 A r t

cardリーダーによってホワイトレベルを校正するため使用される。アクティブ領域は、データブロック、例えば、1104の配列であり、各データブロックは、8個の白色ドット、例えば、1106のギャップによって隣のデータブロックから分離されている。印刷解像度に依存して、代替Art card上のデータブロック数は変化する。1600 dpiの代替Art card上で、配列は8×8でもよい。各データブロック1104は、627×394ドットの次元を有する。ブロック間ギャップ1106が8白色ドットである場合、代替Art cardのアクティブエリアは、5072×3208ドット（1600 dpiのとき、8.1mm×5.1mm）である。

【0478】

データブロック

図49を参照すると、単一のデータブロック1107が示されている。代替Art cardのアクティブ領域は、同一構造のデータブロック1107の配列により構成される。各データブロックの構造は、クロックマーク1109によって囲まれたデータ領域1108と、境界1110と、Target1111である。データ領域は、符号化データだけを含み、クロックマーク、境界、及びターゲットは、特に、データ領域の位置検出を補助し、領域内からのデータの正確な復元を保証するために設けられる。

【0479】

各データブロック1107の次元は、627×934ドットである。この中で、595×384ドットの中央エリアはデータ領域1108である。周囲ドットは、クロックマーク、境界、及びターゲットを保持するため使用される。

【0480】

境界及びクロックマーク

図50はデータブロックを示し、図51及び図52は、そのデータブロックの拡大エッジ部分を示す。図51及び図52に示されるように、二つの5ドット高の境界及びクロックマーク領域1170及び1177が各データブロックに存在し、一方はデータ領域の上方に、もう一方はデータ領域の下側にある。例えば、上端の5ドット高領域は、（データブロックの長辺を引き延ばす）外側黒色ドット境界線1112と、（境界線が独立していることを保証する）白色ドット分離線1113と、3ドット高のクロックマークの組1114と、により構成される。クロックマークは、白色及び黒色行の間で交互に代わり、黒色ブロックの両端から8番目の列で黒色クロックマークから始まる。クロックマークドットとデータ領域内のドットとの間は分離されていない。

【0481】

クロックマークは、代替Art cardが180度回転させて挿入された場合に、同じ相対的な境界／クロックマーク領域が出現するという点で対称性がある。境界1112、1113は、データがデータ領域から読み出されるときに、垂直方向に追従するため、代替Art cardが使用することを意図されている。クロックマーク1114は、データがデータ領域から読み出されるときに、水平方向に追跡することが意図されている。白色のドット直線による境界とクロックマークの分離は、読み取り中にブラー効果が発生する結果として望ましい。境界は、このようにして、両側が白色の黒色直線になり、読み出し時の周波数応答の改良に役立つ。白色と黒色の間で交互に代わるクロックマークは、垂直次元ではなく、水平次元である点を除いて、類似した結果を生ずる。任意の代替Art cardは、追跡のために使用することを意図しているならば、クロックマーク及び境界の位置を見つけなければならない。次のセクションは、クロックマーク、境界及びデータへの方向を示すため設計されたターゲットを取り扱う。

【0482】

ターゲット領域内のターゲット

図54に示されるように、二つの15ドット幅のターゲット領域1116及び1117が各データブロックに存在する。一方はデータ領域の左側に存在し、他方はデータ領域の右側に存在する。ターゲット領域は、方位のため使用される1列のドットによってデータ領域から分離される。ターゲット領域1116及び1117の目的は、クロックマーク、境

10

20

30

40

50

界、及びデータ領域までの進み方を指定することである。各ターゲット領域は、6個のターゲット、例えば、1118を含み、6個のターゲットは、代替Artcardリーダーによって簡単に検出できるように設計される。次に、図53を参照すると、単一Target1120の構造が示されている。各Target1120は、15×15ドットの黒色正方形であり、中心構造1121とランレングス符号化ターゲット番号1122と、を含む。中心構造1121は、単純な白色十字であり、ターゲット番号コンポーネント1122は、単純な2列の白色ドットであり、各列はターゲット番号の各部分に対し2ドットの長さである。したがって、ターゲット番号1のターゲットidの1122は2ドットの長さであり、ターゲット番号2のターゲットidの1122は4ドット幅であり、以下同様に続く。

【0483】

図54に示されるように、ターゲットは、カード挿入に関して回転不変性を有するように配置される。即ち、左のターゲットと右のターゲットは、180度回転している点を除いて同じである。左のターゲット領域1116において、ターゲットは、Target1から6が上端から下端へそれぞれ位置決めされるように配置される。右のターゲット領域において、ターゲットは、ターゲット番号1から6が下端から上端へ位置決めされるように配置される。ターゲット番号idは、常に、データ領域までの半分のところにある。図54の拡大部分図は、右のターゲットが、180度回転している点を除いて、左のターゲットと同一である様子を明瞭に示している。

【0484】

図55に示されるように、Target1124及び1125は、特に、中心が55ドット離されてターゲット領域内に配置される。その上、Target1(1124)の中心から上部クロックマーク領域の第1のクロックマークドット1126までの距離は55ドットであり、ターゲットの中心から下部クロックマーク領域の第1のクロックマークドット（図示せず）までの距離は55ドットである。両方の領域の第1の黒色クロックマークは、ターゲット中心とちょうど一致して始まる（8番目のドット位置は15ドット幅のターゲットの中心である）。

【0485】

図55の略構成図には、ターゲット中心の間の距離と、上部境界／クロックマーク領域内オンTarget1(1124)から第1の黒色クロックマーク(1126)の第1のドットまでの距離が示されている。上部及び下部の両方のターゲットからクロックマークまでの距離は55ドットであり、代替Artcardの両側は対称性があるため（180度回転しているため）、カードは、左から右へ、又は右から左へ読み出される。読み出し方向とは無関係に、方位はデータ領域からデータを抽出するため決定する必要がない。

【0486】

方位列

図56に示されるように、各データブロックには2個の1ドット幅の方位列1127及び1128が存在する。一方はデータ領域の直ぐ左側にあり、他方はデータ領域の直ぐ右側にある。方位列は、代替Artcardリーダーに方向情報を与えるため設けられる。（左ターゲットの右にある）データ領域の左側には、単一の白色ドット列1127が隣接する。（右ターゲットの左にある）データ領域の右側には、単一の黒色ドット列1128が隣接する。ターゲットは回転不変性があるので、これらの二つのドット列によって、代替Artcardリーダーは、代替Artcardの方向、即ち、カードが正しく挿入されたか、又は逆向きに挿入されたかを決定できるようになる。代替Artcardの観点からは、ドットの劣化が無いと仮定すると、次の二つの可能性が存在する。即ち：

＊データ領域の左側のドット列が白色であり、データ領域の右側の列が黒色であるならば、リーダーは、カードが書き込まれたときと同じように挿入されたことを認識する；

＊データ領域の左側のドット列が黒色であり、データ領域の右側の列が白色であるならば、リーダーは、カードが後ろ向きに挿入され、データ領域がそれに応じて回転していることを認識する。リーダーは、代替Artcardから正しく情報を復元するため、適切な動作を選択しなければならない。

【0487】

データ領域

図57に示されるように、データブロックのデータ領域は、各列が384ドットを含む595列により構成され、全部で228480ドットがある。これらのドットは、原データを得るために、解釈され、復号化される。各ドットは単一ビットを表現するので、228480ドットは228480ビット、即ち、28560バイトを表現する。各ドットの解釈は以下の通りである。

【0488】

【表50】

黒 色	1
白 色	0

しかし、ドットから得られたビットの実際の解釈は、原データから代替Art cardのデータ領域内のドットへのマッピングの理解が必要である。

【0489】

原データのデータ領域ドットへのマッピング

次に、最大サイズが910082バイトの原データファイルを取得し、それを、1600dpi代替Art card上の64個のデータブロックのデータ領域内のドットへマッピングするプロセスを説明する。代替Art cardリーダーは、代替Art card上のドットから原データを抽出するためこのプロセスを逆転するであろう。一見して、データをドットにマッピングすることは簡単なことに思われ、即ち、2値データは1と0により構成されるので、黒色ドット及び白色ドットを簡単にカードに書き込むことができるであろう。しかし、この方式は、インクがフェイドし、カードの一部がごみ、汚れ、或いは、場合によっては傷によって損なわれることを許容しない。誤り検出符号化を行わない限り、カードから取得されたデータが正確であるかどうかを検出する方法は無い。更に、冗長符号化を行わない限り、検出された誤りを正す方法はない。したがって、マッピングプロセスの目的は、データ復元を非常に確実に行之、代替Art cardリーダーに、データを正しく読み出したことを認識する能力を与えることである。

【0490】

原データファイルをデータ領域ドットへマッピングするために、以下の三つの基本的なステップが関係する：

＊原データを冗長符号化する；

＊局部的に生じた代替Art cardの損傷の影響を低減するため、符号化データをシ決定論的にシャッフルする；

＊シャッフルされた符号化データをドットとして代替Art cardのデータブロックに書き込む。

【0491】

これらの各ステップは以下のセクションで詳細に考察される。

【0492】

リード・ソロモン符号化を使用する冗長符号

データの代替Art cardへのマッピングは、冗長符号化を利用する方法に大きく依存している。バースト誤りを取り扱い、最小限の冗長性で効率的に誤りを検出し訂正する能力を備えているリード・ソロモン符号化を選択する方が好ましい。リード・ソロモン符号化は、Wicker,

S. と Bhargava,

10

20

30

40

50

V. による''Reed-Solomon Codes and their Application'', IEEE Press, 1994、Rorabough, Cによる''Error Coding Cookbook'', McGraw-Hill, 1994、及びLypkens, H. による''Reed-Solomon Error Correction'', De. Dobb's Journal, Volume 22, Issue 1, January 1997のような標準的なテキストで適切に説明されている。

10

【0493】

リード・ソロモン符号化用の様々なパラメータを使用することが可能であり、異なるシンボルサイズ及び異なる冗長性レベルが含まれる。好ましくは、以下の符号化パラメータ、
 $m = 8$
 $t = 64$
 が使用される。

【0494】

$m = 8$ とすることは、シンボルサイズが8ビット（1バイト）であることを意味する。また、リード・ソロモン符号化されたブロックサイズ n は255バイト（ $2^8 - 1$ シンボル）であることを意味する。最大で t 個のシンボルの訂正を行うため、最終的なブロックサイズで $2t$ 個のシンボルが冗長シンボルを用いて採用されなければならない。 $t = 64$ とすることは、誤りがある場合に、1ブロック毎に64バイト（シンボル）を訂正可能であることを意味する。255バイトの各ブロックは、128（ 2×64 ）冗長バイトを有し、残りの127バイト（ $k = 127$ ）は原データを保持するため使用される。このようにして、
 $n = 255$
 $k = 127$
 である。

20

【0495】

実際の結果では、127バイトの原データが符号化され、255バイトブロックのリード・ソロモン符号化データが得られる。符号化された255バイトブロックは、代替Artcard上に記憶され、その後、代替Artcardリーダーの復号化によって元の127バイトに戻される。データブロックのデータ領域の単一系列内の384ドットは、48バイト（ $384 / 8$ ）を保持し得る。595列は28560バイトを保持し得る。これは、112個のリード・ソロモンブロックに達する（各ブロックは255バイトを含む）。完全な代替Artcardの64個のデータブロックは、全部で7168個のリード・ソロモンブロック（1リード・ソロモンブロック当たり255バイトのとき、 1827840 バイト）を保持し得る。7168個のリード・ソロモンブロックのうちの2個は、制御情報のため確保されるが、残りの7166個はデータを記憶するため使用される。各リード・ソロモンブロックは127バイトの実際のデータを保持し、代替Artcard上に記憶できるデータの総量は910082バイト（ 7166×27 ）である。原データがこの総量未満である場合、データは、正確な個数のリード・ソロモンブロックに適合させるため、符号化することができ、次に、符号化されたブロックは、7166個の全てが使用されるまで複製可能である。図58は利用される符号化の全体的な形態の説明図である。

30

40

【0496】

2個の制御ブロック1132及び1133の各々は、残りの7166個のリード・ソロモンブロックを復号化するために必要な同じ符号化情報、即ち、フルメッセージ（16ビット格納ロー／ハイ）中のリード・ソロモンブロックの個数と、メッセージ（8ビット）の最後のリード・ソロモンブロック内のデータバイトの個数と、

50

を収容する。

【0497】

これらの2個の数は、32回繰り返され(96バイトを消費)、残りの31バイトは確保され、0にセットされる。各制御ブロックは、次に、リード・ソロモン符号化され、127バイトの制御情報は255バイトのリード・ソロモン符号化データに変換される。

【0498】

制御ブロックは、生き残る可能性を高くするため2回記憶される。その上、制御ブロック内のデータの繰り返しは、リード・ソロモン符号を使用する場合に特に重要である。間違いの無いリード・ソロモン符号化ブロックにおいて、データの最初の127バイトは、正しく原データであり、もし、制御ブロックが復号化に失敗したとき(64個を超えるシンボルに誤りがあるとき)、原メッセージを復元するために参照することができる。このようにして、制御ブロックが復号化に失敗した場合、2個の復号化パラメータの最も確からしい値を決定するために、3バイトのセットを検査することができる。復元可能であることは保証されないが、冗長性によって可能性は高くなる。例えば、制御ブロックの最後の159バイトが破壊され、最初の96バイトは完全に大丈夫である。最初の96バイトを参照すると、数字の繰り返しセットが見つかる。これらの数字は、残りの7166個のリード・ソロモンブロックにあるメッセージの残りの部分を復号化するため巧く利用することができる。

【0499】

一例として、正確に9967バイトのデータを含むデータファイルを想定する。必要なリード・ソロモンブロックの数は79個である。最初の78個のリード・ソロモンブロックは完全に利用され、9906バイト(78×127)が使用される。79番目のブロックには、61バイトのデータしか含まれない(残りの66バイトは全て0である)。

【0500】

代替Artcardは、7168個のリード・ソロモンブロックにより構成される。最初の2ブロックは制御ブロックであり、次の79ブロックは符号化データであり、次の79ブロックは符号化データの複製であり、次の79ブロックは符号化データの別の複製であり、以下同様に続く。79個のリード・ソロモンブロックを90回記憶した後、残りの56個のリード・ソロモンブロックは、79ブロックの符号化データのうちの最初の56ブロックの別の複製である(符号化データの最後の23ブロックは、代替Artcard上に十分な場所が存在しないので、これ以上記憶されない)。リード・ソロモン符号化される前の各制御ブロックデータ内の127バイトの16進表現が図59に示されている。

【0501】

符号化データのスクランブル処理

全ての符号化ブロックがメモリ内に隣接して格納されているならば、最大で1827840バイトのデータ(2個の制御ブロックと、7166個の情報ブロックとを合わせて全部で7168個のリード・ソロモン符号化ブロック)を代替Artcard上に記憶することができる。好ましくは、データは、この段階で代替Artcard上にそのまま記憶されるのではなく、即ち、1個のリード・ソロモンブロックの255バイトの全てがカード上で物理的に一体として記憶されるのではない。カードに物理的な破損を生じさせるごみ、汚れ、又は歪みは、一つのリード・ソロモンブロック内の64バイト以上を破壊する可能性があり、そのブロックは復元できなくなる。リード・ソロモンブロックの複製が無い場合、代替Artcardの全体を復号化し得ないことになる。

【0502】

これに対するソリューションは、代替Artcard上に多数のバイトが存在し、代替Artcardは合理的な物理サイズを有するという点を活かすことである。したがって、データはスクランブル処理され、一つのリード・ソロモンブロックからのシンボルが相互に接近しないことが保証される。勿論、カードが劣化する異状ケースでは、リード・ソロモンブロックを復元することは不可能であるが、平均的に、データのスクランブル処理によってカードを非常に頑強にすることができる。選択されたスクランブル方式は単純であ

り、図14に概略的に示されている。リード・ソロモンブロックからのバイト0の全ては、まとめて配置され(1136)、次に、全てのバイト1等も同様である。したがって、7168個のバイト0と、7168個のバイト1等が存在する。代替Art card上の各データブロックは28560バイトを記憶することができる。この結果として、代替Art card上の64個のデータブロックの各々に、各リード・ソロモンブロックからの約4バイトが収容される。

【0503】

このスクランブル方式の下では、代替Art card上の16個のデータブロック全体に対する完全なダメージによって、1リード・ソロモンブロック毎に64個のシンボル誤りが生じる。即ち、代替Art cardにその他のダメージが無い場合、たとえデータの複製が無くても、全てのデータは完全に復元可能である。

10

【0504】

スクランブル処理された符号化データを代替Art cardへ書き込み
原データがリード・ソロモン符号化され、複製され、スクランブル処理されると、1827840バイトのデータを代替Art cardに記憶する必要がある。代替Art card上の64個のデータブロックの各々は、28560バイトを記憶する。

【0505】

データは、単純に代替Art cardのデータブロックへ書き出されるので、第1のデータブロックはスクランブル付きデータのうちの最初の28560バイトを格納し、第2のデータブロックは次の28560バイトを格納し、以下同様に続く。

20

【0506】

図61に示されるように、データブロック内で、データは列に関して左から右へ書き込まれる。このため、データブロック内の最も左側の列は、28560バイトのスクランブル付きデータのうちの最初の48バイトを格納し、最後の列は、28560バイトのスクランブル付きデータのうちの最後の48バイトを格納する。列内では、バイトは、ビット7から始めて、同時に1ビットずつ、上から下へ書き込まれ、ビット0で終了する。ビットがセット(1)されている場合、黒色ドットが代替Art cardに配置され、ビットがクリア(0)されている場合、ドットは配置されず、カードの白色背景カラーのまま残される。

【0507】

例えば、1827840バイトのデータの組は、代替Art cardに記憶されるべき7168個のリード・ソロモン符号化ブロックをスクランブル処理することによって作成できる。最初の28560バイトのデータは最初のデータブロックに書き込まれる。最初の28560バイトのうちの最初の48バイトはデータブロックの最初の列に書き込まれ、次の48バイトは次の列に書き込まれ、以下同様に続く。28560バイトのうちの最初の2バイトが16進のD3

30

5Fである場合を想定する。バイト0のビット7が最初に格納され、次にビット6が格納され、以下同様に続く。次に、バイト1のビット7からバイト1のビット0までが格納される。各「1」は黒色ドットとして格納され、各「0」は白色ドットとして格納されるので、これらの2バイトは、代替Art card上では、以下のドットの組として表現され、
*D3(1101 0011)は、黒、黒、白、黒、白、白、黒、黒になり、
*5F(0101 1111)は、白、黒、白、黒、黒、黒、黒、黒になる。

40

【0508】

代替Art cardの復号化

このセクションは、代替Art cardから正確かつ確実な形で原データを抽出する処理を取り扱う。特に、前の章で説明した代替Art cardのフォーマットを前提として、代替Art cardから元の予め符号化されたデータを抽出する方法を説明する。

【0509】

代替Art cardを復号化する前提の一部として多数の一般的な考察が存在する。

【0510】

50

ユーザ

代替Art cardの目的は、様々なアプリケーションで使用するためのデータを保存することである。ユーザは、代替Art cardを代替Art cardリーダーに挿入し、データが「妥当な時間」内にロードされることを期待する。ユーザの観点からは、モーター輸送が代替Art cardを代替Art cardリーダーへ動かす。これは、問題となる遅れとしては認識されない。なぜならば、代替Art cardは動いているからである。代替Art cardが停止した後の時間は遅延として認識され、代替Art card読み出し方式において最低限に短縮されるべきである。理想的な状態では、代替Art cardの全体が移動中に読み出され、これにより、カードの移動が停止した後に遅延が認識されない。

10

【0511】

好適な実施例の目的のため、代替Art cardを物理的にロードするための妥当な時間は1.5秒に決める。代替Art cardの移動が停止した後の付加的な復号化のための時間を最低限に短縮すべきである。代替Art cardのアクティブ領域は代替Art cardの表面の大部分を覆うので、時間的な関心をその領域に限定することができる。

【0512】

ドットのサンプリング

代替Art card上のドットは、ナイキストの定理(Nyquist's Theorem)を充たすため、少なくとも印刷された解像度の2倍でCCDリーダー等によってサンプリングしなければならない。実際には、これよりも高いレートでサンプリングする方がよい。代替Art cardリーダーの環境では、ドットは、好ましくは、各次元に関してドットが印刷された解像度、即ち、単一ドットを定義するために9画素を必要とする解像度の3倍の解像度でサンプリングされる。代替Art cardのドットの解像度が1600dpiである場合、代替Art cardリーダーのイメージセンサは、4800dpiで画素を走査しなければならない。勿論、ドットがイメージセンサと正確に位置合わせされていない場合、図62に示されるような最悪の最も起こり易いケースでは、ドットは4×4画素エリアに亘って検出されるであろう。

20

【0513】

サンプリングされる各画素は1バイト(8ビット)である。各画素の下位2ビットは、有意なノイズを含む可能性がある。したがって、復号化アルゴリズムは、耐ノイズ性が要求される。

30

【0514】

位置合わせ／回転

ユーザが代替Art cardを回転なしに完全に位置合わせされた状態で代替Art cardリーダーに挿入する可能性は極めて低い。リーダーの入口及びモーター輸送のグリップにおけるある種の物理的制約は、挿入された後、代替Art cardがCCDに対して挿入時の最初の角度で保たれることを保証する。好ましくは、この回転角度は、図63に示されるように最大で1度である。読み出しプロセス中にジッタ及びモーターのガタによって角度が僅かに外れる可能性はあるが、基本的に1度の限界内に収まると考えられる。

【0515】

代替Art cardの物理的寸法は86mm×55mmである。1度の回転は、86mmがCCDの下を通過したとき、カードの実効高さを1.5mm(86sin1°)だけ長くするので、必要なCCD長に影響を与える。

40

【0516】

代替Art cardの読み出しに対する1度の回転の影響は、CCDからの1本の走査線が代替Art cardからの多数の異なるドット列を含むことである。これは、画素の列に対するドットのドリフトを表した図63に拡大された形態で示されている。同図では誇張されているが、実際のドリフトは、57画素毎に最大で1画素列になる。

【0517】

代替Art cardが回転していないとき、単一のドット列は3本の画素走査線によって

50

読むことができる。代替 A r t c a r d の回転が大きくなると、局所的な影響が増加する。読み出されるドットの数が増加すると、回転の影響が加わる時間が長くなる。これらの要因のうちのいずれかが増大すると、代替 A r t c a r d 上の単一列から所定のドットの組を得るために読み出す必要がある画素走査線の数が増加する。以下の表は、特定の代替 A r t c a r d 構造において単一のドット列のために必要な画素走査線の本数を示している。

【0518】

【表51】

領 域	高さ	0°回転	1°回転
アクティブ領域	3208ドット	3 画素列	168 画素列
データブロック	394ドット	3 画素列	21 画素列

10

代替 A r t c a r d の全体を読むため、87mm（86mmと1°の回転による1mmの和）を読む必要がある。4800dpiの場合、これは、16252列の画素列に相当する。

【0519】

C C D（又はその他のリニアイメージセンサ）の長さ

C C D自体の長さは、

－代替 A r t c a r d の物理的高さ（55mm）

－物理的な代替 A r t c a r d の挿入時の垂直方向動き（1mm）

－最大1°の挿入時回転（ $86 \sin 1^\circ = 1.5 \text{ mm}$ ）

に適合しなければならない。

20

【0520】

これらの要因を組み合わせると、全体の長さは57.5mmになる。

【0521】

代替 A r t c a r d リーダーの代替 A r t c a r d イメージセンサ C C D が 4800dpi で走査するとき、1本の走査線は10866画素である。簡単のため、この数字を11000画素に切り上げる。代替 A r t c a r d のアクティブ領域の高さは3208ドット、即ち、9624画素である。データ領域の高さは384ドット、即ち、1152画素である。

30

【0522】

D R A M サイズ

代替 A r t c a r d の読み出し及び復号化に必要なメモリ量は、理想的には最低限に削減される。典型的な代替 A r t c a r d リーダーの配置は、内蔵型システムであり、その場合、メモリリソースは貴重である。このことは、回転の影響によって更に重大な問題になる。上述の通り、代替 A r t c a r d の回転が大きくなると、原ドットを効率的に復元するために要する走査線の数が増加する。

【0523】

アルゴリズムの複雑さと、ユーザの認知する遅延と、頑強性と、メモリ使用量との間にはトレードオフがある。最も簡単なリーダーアルゴリズムの一つは、単に代替 A r t c a r d の全体を走査し、次に、リアルタイムの制約無しに全データを処理することであろう。これは、大規模のメモリ容量を必要とするだけではなく、代替 A r t c a r d の読み出しプロセスと並行に行われるリーダーアルゴリズムよりも長時間を要するであろう。

40

【0524】

代替 A r t c a r d の読み出し及び復号化のために実際に必要なメモリ量は、符号化されたデータを、少量のスクラッチ空間（1から2KB）と共に保持するために必要なスペースの量の2倍である。1600dpiの代替 A r t c a r d の場合、これは、4MBのメモリ必要量を意味する。実際のメモリ使用量は以下のアルゴリズムの説明で詳述される。

【0525】

50

転送レート

D R A M帯域幅の前提条件は、タイミングを考慮するために行う必要があり、特に、代替 A r t c a r dリーダーは、典型的に、内蔵型システムの一部であるので、アルゴリズム設計にある程度の影響を与える。

【0526】

R a m b u s I n cの''D i r e c t R a m b u s

T e c h n o l o g y D i s c l o s u r e'', O c t 1997に記載されているような標準的なR a m b u s

D i r e c t (ラムバスダイレクト) R D R A Mアーキテクチャが想定され、ピークデータ転送レートは1.6GB/秒である。75%の効率(容易に実現できる)を想定すると、平均で1.2GB/秒のデータ転送レートが得られる。したがって、16バイトのブロックにアクセスするために要する平均時間は12nsである。

10

【0527】

汚染データ

物理的に破損した代替A r t c a r dがリーダーに挿入されることがあり得る。代替A r t c a r dは、傷を付けられるか、汚れ若しくはごみによって歪みを生じるかもしれない。代替A r t c a r dリーダーが全てを完全に読み出すことは想定できない。汚染データの影響は、ブラー効果によって更に悪化させられる。なぜならば、汚染データが周囲のきれいなドットに影響を与えるからである。

20

【0528】

ブラー効果環境

ブラー効果は、次の2種類の経路で代替A r t c a r d読み取り環境に組み込まれる：*代替A r t c a r dからC C D間での距離の特性を原因とする自然的なブラー効果；

*代替A r t c a r dの歪み。

【0529】

代替A r t c a r d像の自然的なブラー効果は、C C Dから検出されたデータのオーバーラップが存在するときに生じる。ブラー効果は役に立つ場合がある。なぜならば、ブラー効果は、検出データに高周波数が存在しないこと、及びC C Dによって失われたデータが存在しないことを保証するからである。しかし、C C D画素によって覆われたエリアが非常に広い場合、非常に多くのブラー効果が現れ、データを復元するために要するサンプリングの条件が充たされない。図64は検出データの重なり合いの略説明図である。

30

【0530】

別の形態のブラー効果は、代替A r t c a r dが熱ダメージのために僅かに歪むときに生じる。垂直方向の次元に歪みがあるとき、代替A r t c a r dとC C Dの間の距離は一定ではなく、ブラー効果のレベルはそのエリアの中で変化する。

【0531】

黒色ドット及び白色ドットは、代替A r t c a r dがブラー効果のある読み出し環境で最良のダイナミックレンジを与えるように選択された。ブラー効果は、所与のドットが黒色であるか、又は白色であるかを決定するときに問題を生じる。

40

【0532】

ブラー効果が増大すると、所与のドットは周辺ドットから受ける影響が大きくなる。その結果として、特定のドットのダイナミックレンジは減少する。白色ドット及び黒色ドットの各々が起こり得るあらゆるドットの組によって囲まれる場合を考える。9個のドットにブラー効果が現れ、中央ドットがサンプリングされる。図65は、黒色ドット及び白色ドットに対して得られた中央ドット値の分布の説明図である。

【0533】

この図は、典型的なブラー効果を意図している。0から約180までの曲線1140は黒色ドットのレンジを示す。75から250までの曲線1141は白色ドットのレンジを示す。しかし、ブラー効果が大きくなると、2本の曲線はレンジの中央の方へ向かってシフトし、交わるエリアが増大し、所与のドットが黒色であるか、白色であるかを判定するこ

50

とがより困難になる。交わりの中心点での画素値は曖昧であり、ドットが黒色である可能性と白色である可能性は等しい。

【0534】

ブラー効果が増大すると、読み出しビット誤りの確率が増加する。都合良く、リード・ソロモン復号化アルゴリズムは、 t 個のシンボル誤りまで正常にこれらに対処することができる。図65は、特定のシンボル誤り率の場合に復元できない代替Art cardのリード・ソロモンブロックの予測数のグラフである。リード・ソロモン復号化方式が優れた性能を示し、次に、実質的に低下することに注意すべきである。リード・ソロモンブロックの複製が無い場合、1ブロックに誤りが生じるだけでデータを復元できなくなる。勿論、ブロック複製を行った場合、代替Art card復号化の機会が増加する。

10

【0535】

図66は、誤りのあるリード・ソロモンブロックの個数に対応するシンボル（バイト）誤りだけを示している。対処可能なブラー効果の量と、カードに生じたダメージの量との間にはトレードオフが存在する。全ての誤り検出及び訂正はリード・ソロモンデコーダによって行われるので、リード・ソロモンデータブロック毎に対処可能な誤りの個数は有限である。ブラー効果によって取り込まれる誤りが増加すると、代替Art cardのダメージによる誤りの中で対処可能な誤りの個数は減少する。

【0536】

代替Art card復号化の概要

上述の通り、ユーザが代替Art cardを代替Art card読み取りユニットに挿入するとき、モーター輸送は、理想的には、代替Art cardを運び、モノクロリニアCCDイメージセンサの傍らを通させる。カードは次元毎に印刷された解像度の3倍の解像度でサンプリングされる。代替Art card読み取りハードウェア及びソフトウェアは、1度までの回転と、モーター輸送によって生じるジッタ及び振動と、代替Art cardからCCDまでの距離の変動によって生じるブラー効果を補正する。データのデジタルビットイメージが、以下に説明される複雑な方法によって、サンプリングされた像から抽出される。リード・ソロモン復号化は、代替Art card上の生データの25%までの任意に分布したデータコラプションを訂正する。約1MBの訂正データが1600dpiカードから抽出される。

20

【0537】

復号化に関連するステップは図67に示されるようなステップである。

30

【0538】

復号化プロセスに必要なステップは、

- *印刷された解像度の3倍の解像度で代替Art cardを走査する（例えば、4800dpiで1600dpiの代替Art cardを走査する）ステップ1144と、
 - *カード上で走査されたドットからデータビットマップを抽出するステップ1145と、
 - *代替Art cardが逆向きに挿入された場合に、ビットマップを反転するステップ1146と、
 - *符号化データをスクランブル解除するステップ1147と、
 - *ビットマップからのデータをリード・ソロモン復号化するステップ1148と、
- である。

40

【0539】

アルゴリズムの概要

フェーズ1ーリアルタイムExtractingBitImage

利用可能なメモリ（4MB）と、1600dpiの代替Art cardの走査された全画素を保持するための必要メモリ（172.5MB）との簡単な比較によって、カードが複数回読み出されない限り（現実的な選択肢ではない）、画素データからのビットマップの抽出は、代替Art cardがCCDの傍を通させられる間に、オンザフライ方式でリアルタイムに実行する必要がある。このフェーズでは、二つのタスク、即ち、

- *代替Art cardを4800dpiで走査するタスク

50

＊カード上で走査されたドットからデータビットマップを抽出するタスク
を実行しなければならない。

【0540】

ビットイメージの回転及びスクランブル解除は、ビットイメージ全体が抽出されるまで行えない。したがって、抽出されたビットイメージを保持するためにメモリ領域を割り当てる必要がある。ビットイメージは、2 MBの範囲内で容易に収容され、抽出プロセスで使用するための2 MBが確保される。

【0541】

CCDからの現在の走査線だけを調べる間にビットイメージを抽出するのではなく、窓として機能するバッファを代替Artcardに割り付け、最新のN本の走査線読み出しを記憶することが可能である。メモリ必要量は、代替Artcardの全体がこのようにして記憶されることは許容しないが（172.5 MBが必要になるであろう）、190画素列を記憶するため2 MBを割り付けることは（各走査線は11000バイト以上を必要としない）、ExtractingBitImageプロセスを単純化させる。

【0542】

したがって、4 MBメモリの使用法は以下の通りである：

＊抽出されたビットイメージのための2 MB；

＊走査された画素のための～2 MB；

＊フェーズ1のスクラッチデータのために1.5 KB（アルゴリズムによる要求次第）。

【0543】

フェーズ1に要する時間は1.5秒である。なぜならば、この時間は、代替ArtcardがCCDの傍を通過し、物理的にロードされるために要する時間であるからである。

【0544】

フェーズ2ービットイメージからのデータ抽出

ビットイメージが抽出されると、ビットイメージはスクランブル解除され、場合によっては180°回転させることが必要である。ビットイメージは次に復号化される。フェーズ2は、Artcardの動きが停止している点でリアルタイム必要条件が無く、ユーザが時間経過を認識することだけが問題である。したがって、フェーズ2は、代替Artcardを復号化する残りのタスクに関係する。即ち、

＊ビットイメージを再編成し、代替Artcardが逆向きに挿入された場合には反転する；

＊符号化されたデータをスクランブル解除する；

＊ビットイメージからのデータをリード・ソロモン復号化する。

【0545】

フェーズ2への入力は2 MBビットイメージバッファである。スクランブル解除及び回転は、その場では実行し得ないので、第2の2 MBバッファが必要である。フェーズ1で走査された画素を保持するため使用される2 MBバッファは不要になっているので、回転させられたスクランブル無しデータを記憶するため使用可能である。

【0546】

リード・ソロモン復号化タスクは、スクランブル無しのビットイメージを取得し、それを910082バイトに復号化する。復号化は元の位置で実行可能であるが、指定された別の場所で実行してもよい。復号化プロセスは、補助的なメモリバッファを必要としない。

【0547】

したがって、4 MBのメモリの使用法は以下の通りである：

＊（フェーズ1からの）抽出されたビットイメージのための2 MB；

＊スクランブル無しの、回転された可能性のあるビットイメージのための～2 MB；

＊フェーズ2のスクラッチデータのための<1 KB（アルゴリズムによる要求次第）。

【0548】

フェーズ2に要する時間はハードウェアに依存し、リード・ソロモン復号化のために要する時間によって制限される。LSIロジックのL64712のような専用コア、又は等価的なC

10

20

30

40

50

P U / D S P の組み合わせを使用することにより、フェーズ 2 は 0.32 秒を要することが推定される。

【0549】

フェーズ 1 - ExtractingBitImage

これは、アルゴリズムのリアルタイムフェーズであり、C C D によって走査されたような代替 A r t c a r d からビットイメージを抽出することに関連する。

【0550】

図 68 に示されるように、フェーズ 1 は、二つの非同期的なプロセスストリームに分解することができる。そのうちの第 1 のストリームは、単に、C C D からの代替 A r t c a r d の画素をリアルタイムで読み出し、その画素を D R A M へ書き込む。第 2 のストリームは、画素を参照し、ビットを抽出する。第 2 のプロセスストリームは、それ自体が二つのプロセスに分解される。第 1 のプロセスは、大域的なプロセスであり、代替 A r t c a r d のスタートの位置検出に関する。第 2 のフェーズは ExtractingBitImage 固有である。

【0551】

図 69 は、データ／プロセスの観点からのデータフロー説明図である。

【0552】

タイミング

1600dpi 代替 A r t c a r d の全体に対して、最大で 16252 画素列を読み出すことが必要である。代替 A r t c a r d の全体に対する合計時間が 1.5 秒である場合、これは、様々なプロセスの経過中に、1 画素列当たりの最大時間が 92296ns であることを意味する。

【0553】

プロセス 1 - C C D からの画素読み出し

C C D は、代替 A r t c a r d を 4800dpi で走査し、1 列当たりに 11000 個の 1 バイト画素を生成する。このプロセスは、C C D からデータを取得し、それを D R A M へ書き込むだけであり、D R A M から画素データを読み出している他のプロセスからは完全に独立している。図 70 には関連したステップが示されている。

【0554】

画素は、190 列の全画素列を保持し得る 2MB バッファに連続して書き込まれる。バッファは、常に、最近に読み出された 190 列を保持する。その結果として、画素データを読み出そうとするプロセス（例えば、プロセス 2 及びプロセス 3）は、最初に、所定の列が探索する場所を知る必要があり、次に、要求されたデータが実際にバッファ内に存在することが確実であるように十分に高速でなければならない。

【0555】

プロセス 1 は、現在の CurrentScanLine が他のプロセスから利用できるようにさせるので、他のプロセスは、未だ読み出されていない走査線からの画素へのアクセスを試みないことを保証することができる。

【0556】

単一のデータ列（11000 バイト）を D R A M へ書き出すために要する時間は、 $11000 / 16 * 12 = 8256ns$ である。

【0557】

プロセス 1 は、したがって、利用可能な D R A M 帯域幅の 9% 未満（ $8256 / 92296$ ）しか使用しない。

【0558】

プロセス 2 - 代替 A r t c a r d のスタート検出

このプロセスは、走査された代替 A r t c a r d 上のアクティブエリアの位置検出と関連している。このステージへの入力 は D R A M（プロセス 1 によってそこに収容された）からの画素データである。出力は、代替 A r t c a r d 上の最初の 8 個のデータブロックに対する境界の組であり、プロセス 3 への入力として要求される。プロセス 3 の上位レベル概要が図 71 に示されている。

【0559】

代替Artcardは、挿入時に垂直方向の動きが1mmある。1度の回転によって、更に、 $1.5\text{mm} (86 \sin 1^\circ)$ の垂直方向の動きが生じる。その結果として、全部で垂直方向の動きは2.5mmである。1600dpiのとき、これは、約160ドットの動きに相当する。単一のデータブロックの高さは394ドットに過ぎないので、この動きは、データブロックの半分よりも少し少ない。データブロックが存在する場所をより良く推定するためには、代替Artcard自体を検出しなければならない。

【0560】

したがって、プロセス2は、

*代替Artcardのスタートを位置検出する部分と、もし、検出されたならば、
 *代替Artcardのスタートに基づいて最初の8個のデータブロックの境界を計算する部分と、
 を含む。

10

【0561】

代替Artcardのスタート位置検出

代替Artcardエリアの外側で走査された画素は黒色である（表面は黒色のプラスチック、又はその他の非反射型表面でもよい）。代替Artcardエリアの境界は白色である。画素列を1列ずつ処理し、画素を黒色又は白色の何れかに関してフィルタ処理するならば、黒色から白色への変化点は、代替Artcard上のスタートを特徴付ける。最高レベルのプロセスは以下の通りである：

20

```
for (Column=0; Column < MAX_COLUMN; Column++)
{
    Pixel = ProcessColumn(Column)
    if (Pixel)
        return (Pixel, Column)           // success!
}
return failure                          // no alternative Artcard found
```

ProcessColumn関数は単純である。走査された列の二つのエリアからの画素は、それらが黒色又は白色のどちらであるかを判定するため閾値フィルタに通される。一定数の白色画素を待ち、所定の個数が検出されたとき、代替Artcardのスタートを報せることが可能である。画素列を処理するロジックは以下の疑似コードに示されている。列中に代替Artcardが検出されなかった場合、0が返される。さもなければ、検出された位置の画素数が返される：

30

```

// Try upper region first
count = 0
for (i=0; i<UPPER_REGION_BOUND; i++)
{
    if (GetPixel(column, i) < THRESHOLD)
    {
        count = 0 // pixel is black
    }
    else
    {
        count++ // pixel is white
        if (count > WHITE_ALTERNATIVE ARTCARD)
            return i
    }
}

// Try lower region next. Process pixels in reverse
count = 0
for (i=MAX_PIXEL_BOUND; i>LOWER_REGION_BOUND; i--)
{
    if (GetPixel(column, i) < THRESHOLD)
    {
        count = 0 // pixel is black
    }
    else
    {
        count++ // pixel is white
        if (count > WHITE_ALTERNATIVE ARTCARD)
            return i
    }
}

//Not in upper bound or in lower bound. Return failure
return 0

```

データブロック境界の計算

このステージでは、代替 A r t c a r d が検出されている。代替 A r t c a r d の回転に応じて、代替 A r t c a r d の上部又は代替 A r t c a r d の下部の何れかが検出されている。プロセス 2 の第 2 ステップはどちらが検出されたかを判定し、フェーズ 3 のためのデータブロック境界を適切にセットする。

【 0 5 6 2 】

フェーズ 3 を見ると、フェーズ 3 はデータブロックセグメント境界に関して動作することがわかる。各データブロックは、データブロックのデータ領域を位置検出するためにターゲットを探索すべき場所を決定するため、StartPixel 及び EndPixel を含む。

【 0 5 6 3 】

画素値がカードの上半分に存在する場合、それを第 1 の StartPixel 境界としてそのまま使用することができる。画素値がカードの下半分に存在する場合、画素値が最後のセグメン

トのEndPixel境界になるように逆行させることが可能である。代替A r t c a r dのデータサイズの幅で前進又は後退し、各セグメントに適切な境界をセットする。これにより、代替A r t c a r dからのデータ抽出を開始する準備が整う：

```
// Adjust to become first pixel if is lower pixel
```

```
if (pixel > LOWER_REGION_BOUND)
```

```
{
```

10

```
pixel -= 6 * 1152
```

```
if (pixel < 0)
```

```
pixel = 0
```

```
}
```

20

```
for (i=0; i<6; i++)
```

```
{
```

```
endPixel = pixel + 1152
```

```
segment[i].MaxPixel = MAX_PIXEL_BOUND
```

30

```
segment[i].SetBounds(pixel, endPixel)
```

```
pixel = endPixel
```

```
}
```

MaxPixel値はプロセス3で定義され、SetBounds関数は、0及びMaxPixelに対するStartPixel及びEndPixelクリッピングをセットするだけである。

40

【0564】

プロセス3－画素からのビットデータ抽出

これは代替A r t c a r dリーダーアルゴリズムの心臓部である。このプロセスは、CCD画素データからのビットデータ抽出処理に関係する。このプロセスは、本質的に、プロセス2によって作成され、プロセス3によって維持されたスクラッチ情報に基づいて、画素データからビットイメージを作成する。このプロセスの上位レベル概要は図72に示されている。

【0565】

単に代替A r t c a r dの画素列を読み出し、どの画素がどのデータブロックに属しているかを判定するのではなく、プロセス3は、その逆の作用をする。プロセス3は、所定の

50

データブロックの画素を探索すべき場所がわかっている。プロセス 3 は、これを行うため、論理的な代替 Art card を 8 個のセグメントに分割する。各セグメントは、図 7 3 に示されるように 8 個のデータブロックを含む。

【0566】

図示されたセグメントは論理的な代替 Art card と一致する。物理的に、代替 Art card はある量だけ回転させられている可能性が高い。セグメントは、論理的な代替 Art card 構造に固定されたままであるため、回転に依存しない。あるセグメントは、以下の 2 状態のうち的一方をとり得る：

*LookingForTargets：この状態では、このセグメントの正確なデータブロック位置は未だ決定されていない。ターゲットは、セグメント境界によって指定された境界内の画素列データを走査することによって位置検出されている。ターゲット、並びに、黒色及び白色のためにセットされた境界を用いてデータブロックが位置検出されると、状態は、ExtractingBitImage へ変化する；

*ExtractingBitImage：この状態では、データブロックは正確に位置検出されており、データビットは同時に 1 ドット列ずつ抽出され、代替 Art card ビットイメージに書き込まれる。データブロッククロックマークの追跡は、回転とは無関係に正確なドット復元を与えるので、セグメント境界は無視される。データブロック全体が抽出されると、新しいセグメント境界が、現在位置に基づいて次のデータブロックに対して計算される。状態は、LookingForTargets へ変化する。

【0567】

このプロセスは、各領域から 8 個ずつの 64 データブロックの全てが抽出されたときに終了する。

【0568】

各データブロックは 595 のデータ列により構成され、各データ列は 48 バイトである。好ましくは、データブロック用の 2 個の方位列がそれぞれ 48 バイト毎に抽出され、全部で 1 データブロック当たり 28656 バイトが抽出される。簡単のため、2 MB のメモリを 64 × 32 K のチャンクに分割することができる。所定のセグメントの n 番目のデータブロックは、

$$\text{StartBuffer} + (256k \cdot n)$$

で表される場所に格納される。

【0569】

セグメントのデータ構造

8 個のセグメントの各々は、データ構造が関連付けられる。各セグメントを定義するデータ構造は、スクラッチデータエリアに格納される。その構造は以下の表に列挙することができる。

【0570】

10

20

30

【表 5 2】

データ名	コメント
CurrentState	セグメントの現在状態を定義する。 <i>LookingForTargets</i> と <i>ExtractingBitImage</i> のうち的一方をとり得る。初期値は <i>LookingForTargets</i> である。
	<i>LookingForTargets</i> の間に使用される。
StartPixel	セグメントの上側画素境界である。 プロセス 2 によって初期セットされる。
EndPixel	セグメントの下側画素境界である。 プロセス 2 によって初期セットされる。
MaxPixel	走査線のための最大画素数である。各セグメントに対して同一値 : 10866 にセットされる。
CurrentColumn	ターゲット探索中に到達する画素列
FinalColumn	ターゲット探索のために最後の画素列を定義する。
LocatedTargets	位置検出されたターゲットのリストを指示する。
PossibleTargets	ターゲット候補である現在調査中の画素形状を表現するターゲット構造体へのポインタの組を指示する。
AvailableTargets	現在未使用であるターゲット構造体へのポインタの組を指示する。
TargetFound	現データブロック内でこれまでに検出されたターゲット数
PossibleTargetCount	PossibleTargets リスト内の要素数
AvailableTartetCount	AvailableTargets リスト内の要素数
	<i>ExtractingBitImage</i> の間に使用される。
BitImage	次のデータブロックを格納する DRAM 中のビットイメージデータエリアのスタート : セグメント 1 = X、セグメント 2 = X + 32k 等 状態が <i>ExtractingBitImageData</i> から <i>LookingForTargets</i> へ変化する毎に 256k ずつ進む。
CurrentByte	次の抽出バイトを格納するビットイメージ内のオフセット
CurrentDotColumn	現在のクロックマーク／ドット列番号を保持する。 状態が <i>LookingForTargets</i> から <i>ExtractingBitImage</i> へ変化したときに -8 をセットする。
UpperClock	現在のの上側クロックマーク／境界の座標(列／画素)
LowerClock	現在のの下側クロックマーク／境界の座標(列／画素)
CurrentDot	現在ドット列の現在データドットの中心である。データ列の最初(最上部) のドットの中心に初期セットされる。
DataDelta	次のドットの中心へ移動するために加算すべき量 (列／画素)
BlackMax	この値よりも大きい画素値のドットは確実に白色である。
WhiteMin	この値よりも小さい画素値のドットは確実に黒色である。
MidRange	黒色又は白色である可能性が等しい画素値。スマートな手法でドットが判定されなかったとき、この値がドットを決定するため使用される。この値よりも小さい画素は黒色であり、大きい画素は白色である。

10

20

30

40

プロセス 3 の上位レベル

プロセス 3 は、各セグメントに対して繰り返して、セグメントの現在状態に依存して単一ラインの処理を実行するだけである。擬似コードは以下のように簡潔である：

50

```

blockCount = 0
while (blockCount < 64)
    for (i=0; i<8; i++)
    {
        finishedBlock = segment[i].ProcessState()
        if (finishedBlock)
            blockCount++
    }

```

10

20

プロセス 3 は、指定された時間が経過しても終了しない場合には、外部制御プロセスによって停止される必要がある。これは、データが抽出できなかった場合に限られるであろう。簡単な仕組みは、プロセス 1 が代替 A r t c a r d の読み出しを終了した後にカウントダウンを開始する仕組みである。プロセス 3 がその時間までに終了しなかった場合、代替 A r t c a r d からのデータは復元不可能である。

【 0 5 7 1 】

CurrentState = LookingForTargets

ターゲットは、同時に 1 画素列ずつ画素の列を読み出すことによって検出されるのであって、(StartPixelとEndPixelの間にある) 所定の画素のバンド内のドットを検出することによって画素のある種のパターンが検出されるのではない。画素列は、全てのターゲットが検出されるか、又は指定された数の列が処理されるまで、同時に 1 列ずつ処理される。その時点で、ターゲットを処理することが可能であり、データはクロックマークを用いて位置検出される。状態は、データを抽出する時であることを示すためExtractingBitImageに変更される。十分な数の有効ターゲットを位置検出できなかった場合、データブロックは無視され、失敗したデータブロック内の列を確実にスキップし、次のデータブロック内でターゲットを探索するプロセスを再開する。これは以下の擬似コードで表すことができる：

30

```

finishedBlock = FALSE
if(CurrentColumn < Process1.CurrentScanLine)
{
    ProcessPixelColumn()
    CurrentColumn++
}
if ((TargetsFound == 6) || (CurrentColumn > LastColumn))
{
    if (TargetsFound >= 2)
        ProcessTargets()
        if (TargetsFound >= 2)
        {
            BuildClockmarkEstimates()
            SetBlackAndWhiteBounds()
            CurrentState = ExtractingBitImage
            CurrentDotColumn = -8
        }
        else
        {
            // data block cannot be recovered. Look for
            // next instead. Must adjust pixel bounds to
            // take account of possible 1 degree rotation.
            finishedBlock = TRUE
            SetBounds(StartPixel-12, EndPixel+12)
            BitImage += 256KB
            CurrentByte = 0
            LastColumn += 1024
            TargetsFound = 0
        }
    }
    return finishedBlock
}

```

ProcessPixelColumn関数

各画素列は、ターゲットを識別するある種の画素のパターンを探索するため、指定された境界内（StartPixelとEndPixelの間）で処理される。単一ターゲット（ターゲット番号2

）の構造は、既に図 5 4 に示されている通りである。

【0 5 7 2】

画素の観点から、ターゲットは、以下の事項によって識別することができる：

- * 左の黒色領域：これは、多数の隣接した黒色画素により構成された多数の画素列であり、ターゲットの最初の部分を構築する；
- * ターゲット中心：これは、更なる黒色列の中心にある白色領域である；
- * ターゲット番号：これは、黒色で囲まれた白色領域であり、白色領域はその長さによってターゲット番号を定義する；
- * 第 3 の黒色領域：これは、ターゲット番号の後の二つの黒色ドット領域である。

【0 5 7 3】

要求されるプロセスの概要は図 7 4 に示されている。

【0 5 7 4】

識別は黒色又は白色画素だけに基づいているので、各列からの画素 1 1 5 0 は、黒色又は白色を検出するためフィルタ 1 1 5 1 を通され、次に、ランレングス符号化 1 1 5 2 される。ランレングスは、次に、状態機械 1 1 5 3 へ送られ、状態機械は、最後の 3 個のランレングスと、最後から 4 番目の色を取得する。これらの値に基づいて、ターゲット候補は、識別ステージの各々を通過する。

【0 5 7 5】

最大及び最小収集プロセス GatherMin&Max 1 1 5 5 は、セグメントの処理中に見つかった最小値及び最大値を保持するだけである。これらは、ターゲットが位置検出されたとき、BlackMax、WhiteMin、及びMidRange値をセットするため使用される。

【0 5 7 6】

各セグメントは、ターゲットの探索においてターゲット構造体の組を保持する。ターゲット構造体自体はメモリ内を動き回らないが、幾つかのセグメント変数はこれらのターゲット構造体へのポインタのリストを指示する。これらの 3 個のポインタリストを再度示す。

【0 5 7 7】

【表 5 3】

LocatedTargets	位置検出されたターゲットを表現するターゲット構造体の組を指示する。
PossibleTargets	ターゲット候補である現在調査中の画素形状を表現するターゲット構造体へのポインタの組を指示する。
AvailableTargets	現在未使用であるターゲット構造体へのポインタの組を指示する。

これらのリストポインタの各々と関連したカウンタ、即ち、TargetsFound、PossibleTargetCount、及びAvailableTargetCountが存在する。

【0 5 7 8】

代替 A r t c a r d がロードされる前に、TargetsFound及びPossibleTargetCountは 0 にセットされ、AvailableTargetCountは 2 8 （ターゲット境界の最小サイズは 4 0 画素であり、データエリアは約 1 1 5 2 画素であるので、調査中のターゲット構造体候補の最大数）にセットされる。ターゲットポインタレイアウトの一例は図 7 5 に示される。

【0 5 7 9】

新しいターゲット候補が検出されるとき、それらは、AvailableTargetsリスト 1 1 5 7 から取得され、ターゲットデータ構造体は更新され、この構造体へのポインタはPossibleTargetsリスト 1 1 5 8 に追加される。ターゲットが完全に検証されたとき、そのターゲットはLocatedTargetsリスト 1 1 5 9 へ追加される。ターゲット候補が最終的にターゲットではないことが判明した場合、ターゲット候補はAvailableTargetsリスト 1 1 5 7 へ戻される。この結果として、いつでも循環している 2 8 個のターゲットポインタが常に存在し、リストの間を移動している。

【0 5 8 0】

ターゲットデータ構造体 1 1 6 0 は以下の形式をとり得る。

【 0 5 8 1 】

【表 5 4】

データ名	コメント
CurrentState	ターゲット探索の現在状態
DetectCount	ターゲットが所定の状態に入っている長さのカウント
StartPixel	ターゲットはどこでスタートするか?このターゲットの全ての画素のラインはこの画素値の許容範囲内でスタートすべきである。
TargetNumber	ターゲットのターゲット番号を示す (読み出されたものに従う)
Column	ターゲットの中心列座標の最良推定値
Pixel	ターゲットの中心画素座標の最良推定値

10

ターゲット検出モジュール 1 1 6 2 (図 7 4) 内の ProcessPixelColumn 関数は、全てのランレングスを一つずつ調べ、(StartPixelによって)ランを既存のターゲット候補と比較するか、又は今までに知られていないターゲット候補が検出された場合に、新しいターゲット候補を作成する。全てのケースにおいて、比較はS0.colorが白色であり、S1.colorが黒色である場合に限り行われる。

【 0 5 8 2 】

20

次に、ProcessPixelColumnの擬似コードを示す。第 1 のターゲットが肯定的に識別された場合、ターゲットをチェックすべき最後の列は、そこから最大距離の範囲内にあるとして決定される。1 ° の回転がある場合、最大距離は 1 8 画素列である：

```

pixel = StartPixel
t = 0
target = PossibleTarget[t]
while ((pixel < EndPixel) && (TargetsFound < 6))
{
    if ((S0.Color == white) && (S1.Color == black))
    {
        do
        {
            keepTrying = FALSE
            if
            (
                (target != NULL)
                &&
                (target->AddToTarget(Column, pixel, S1, S2, S3))
            )
            {
                if (target->CurrentState == IsATarget)
                {
                    Remove target from PossibleTargets List
                    Add target to LocatedTargets List
                    TargetsFound++
                    if (TargetsFound == 1)
                        FinalColumn = Column + MAX_TARGET_DELTA
                }
                else if (target->CurrentState == NotATarget)
                {
                    Remove target from PossibleTargets List
                    Add target to AvailableTargets List
                    keepTrying = TRUE
                }
            }
        }
    }
}

```

```

else
{
    t++      // advance to next target
}
target = PossibleTarget[t]
}
else
{
    tmp = AvailableTargets[0]
    if (tmp->AddToTarget(Column,pixel,S1,S2,S3)
    {
        Remove tmp from AvailableTargets list
        Add tmp to PossibleTargets list
        t++      // target t has been shifted right
    }
}
} while (keepTrying)
}
pixel += S1.RunLength
Advance S0/S1/S2/S3
}

```

AddToTargetはターゲット検出モジュール内の関数であり、特定のランを所与のターゲットに加算することができるかどうかを判定する：

＊ランがターゲットの開始位置の許容範囲内に入るならば、そのランは現在のターゲットと直接的に関連付けられるので、それに適用することが可能である；

＊ランがターゲットの前で始まる場合、既存のターゲットが依然としてOKであり、そのランとは関係がないと想定する。したがって、ターゲットはそのままの状態に残され、返値FALSEは、ランが適用されなかったことを呼び出し元へ通知する。呼び出し元は、引き続いてランをチェックして、ランがその固有の全く新しいターゲットを開始するかどうかを調べる；

＊ランがターゲットの後で始まる場合、そのターゲットが候補ターゲットではなくなったと考える。状態は、NotATargetに変更され、返値TRUEが返される；

ランをターゲットに適用すべき場合、特別の動作が現在の状態と、S1、S2及びS3のランの組に基づいて実行される。AddToTargetの擬似コードは以下の通りである：

40

```

MAX_TARGET_DELTA = 1
if (CurrentState != NothingKnown)
{
    if (pixel > StartPixel)                // run starts after target
    {
        diff = pixel - StartPixel
        if (diff > MAX_TARGET_DELTA)
        {
            CurrentState = NotATarget
            return TRUE
        }
    }
    else
    {
        diff = StartPixel - pixel
        if (diff > MAX_TARGET_DELTA)
            return FALSE
    }
}
runType = DetermineRunType(S1, S2, S3)
EvaluateState(runType)
StartPixel = currentPixel
return TRUE

```

画素ランのタイプはDetermineRunTypeにおいて以下のように識別される。

【 0 5 8 3 】

【 表 5 5 】

画素ランのタイプ	
タイプ	識別方法(S1 は常に黒色)
TargetBorder	S1 = $40 < \text{RunLength} < 50$ S2 = 白色ラン
TargetCenter	S1 = $15 < \text{RunLength} < 26$ S2 = 白色ラン 但し、 $[\text{RunLength} < 12]$ S3 = 黒色ラン 但し、 $[15 < \text{RunLength} < 26]$
TargetNumber	S2 = 白色ラン 但し、 $[\text{RunLength} \leq 40]$

状態推定手続EvaluateStateは現在の状態及びランタイプに依存して動作する。

【 0 5 8 4 】

その動作は以下に表形式で示される。

【 0 5 8 5 】

【 表 5 6 】

現在状態	画素ランのタイプ	動 作
NothingKnown	TargetBorder	DetectCount = 1 CurrentState = LeftOfCenter
LeftOfCenter	TargetBorder	DetectCount ++ if (DeleteCount > 24) CurrentState = NotATarget
	TargetCenter	DetectCount = 1 CurrentState = InCenter Column = currentColumn Pixel = currentPixel + S1.RunLength
		CurrentState = NotATarget
InCenter	TargetCenter	DetectCount ++ tmp = currentPixel + S1.RunLength if (tmp < Pixel) Pixel = tmp if (DeectctCount > 13) CurrentState = NotATarget
	TargetBorder	DetectCount = 1 CurrentState = RightOfCenter
		CurrentState = NotATarget
RightOfCenter	TargetBorder	DetectCount ++ if (DetectCount > = 12) CurrentState = NotATarget
	TargetNumber	DetectCount = 1 CurrentState = InTargetNumber TargetNumber = (S2.RunLength + 2) / 6
		CurrentState = NotATarget
InTargetNumber	TargetNumber	tmp = (S2.RunLength + 2) / 6 if (tmp > TargetNumber) TargetNumber = tmp DetectCount ++ if (DetectCount > = 12) CurrentState = NotATarget
	TargetBorder	if (DetectCount > = 3) CurrentState = IsATarget else CurretState = NotATarget
		CurrentState = NotATarget
IsATarget 又は NotATarget	—	—

10

20

30

40

50

ターゲット処理

位置検出されたターゲット (LocatedTargetsリスト内) は、位置検出された順序で記憶される。代替 A r t c a r d の回転に依存して、これらのターゲットは、画素昇順又は画素降順である。その上、ターゲットから復元されたターゲット番号は、誤りがあるかもしれない。また、間違ったターゲットを復元したかもしれない。クロックマーク推定量が取得できる前に、ターゲットは、無効ターゲットが確実に廃棄されるように処理する必要がある

り、有効ターゲットは、間違っているならばターゲット番号が修復される（例えば、ごみのために破損されたターゲット番号）。二つの主要なステップが関連している：

＊ターゲットを画素昇順にソートするステップ；

＊誤りのあるターゲット番号を見つけ、修復するステップ。

【0586】

最初のステップは簡単である。ターゲット検索の性質は、データが画素昇順又は画素降順の何れかで常に記憶されるべきことを意味する。簡単なスワップソートは、6個のターゲットが既に正しくソートされているならば、最大で14回の比較がスワップ無しに行われる。データがソートされていない場合、14回の比較は3回の3回のスワップと共に行われる。以下の擬似コードはソーティングプロセスを示す：

10

```
for (i = 0; i < TargetsFound-1; i++)
```

```
{
```

```
    oldTarget = LocatedTargets[i]
```

```
    bestPixel = oldTarget->Pixel
```

```
    best = i
```

```
    j = i+1
```

```
    while (j < TargetsFound)
```

20

```
    {
```

```
        if (LocatedTargets[j]-> Pixel < bestPixel)
```

```
            best = j
```

```
        j++
```

```
    }
```

```
    if (best != i) // move only if necessary
```

30

```
        LocatedTargets[i] = LocatedTargets[best]
```

```
        LocatedTargets[best] = oldTarget
```

```
    }
```

```
}
```

誤ったターゲット番号を位置検出し修復することは、多少複雑になる。順番に、検出されたN個のターゲットの各々が正しいと仮定される。他のターゲットは、この「正しい」ターゲットと比較され、TargetNが正しい場合に変更を必要とするターゲットの個数がカウントされる。変更の回数が0であるならば、全てのターゲットは既に正しい筈である。さもなければ、他のターゲットに対して要求する変更の回数が最も少ないターゲットが変更の基準として使用される。所与のターゲットのターゲット番号及び画素位置が、「正しい」ターゲットの画素位置及びターゲット番号と比較されたときに、相互に関連しないならば、変更が登録される。この変更は、ターゲットのターゲット番号の更新を意味し、或いは、ターゲットの削除を意味する。上昇するターゲットは、（既にソートされているので）昇順の画素を有すると仮定することができる：

40

```

kPixelFactor = 1/(55 * 3)
bestTarget = 0
bestChanges = TargetsFound + 1
for (i=0; i< TotalTargetsFound; i++)
{
    numberOfChanges = 0;
    fromPixel = (LocatedTargets[i])->Pixel
    fromTargetNumber = LocatedTargets[i].TargetNumber
    for (j=1; j< TotalTargetsFound; j++)
    {
        toPixel = LocatedTargets[j]->Pixel
        deltaPixel = toPixel - fromPixel
        if (deltaPixel >= 0)
            deltaPixel += PIXELS_BETWEEN_TARGET_CENTRES/2
        else
            deltaPixel -= PIXELS_BETWEEN_TARGET_CENTRES/2
        targetNumber =deltaPixel * kPixelFactor
        targetNumber += fromTargetNumber
        if
        (
            (targetNumber < 1)||(targetNumber > 6)
            ||
            (targetNumber != LocatedTargets[j]-> TargetNumber)
        )
            numberOfChanges++
    }
    if (numberOfChanges < bestChanges)
    {
        bestTarget = i
        bestChanges = numberOfChanges
    }
    if (bestChanges < 2)
        break;
}

```

多くの場合、この関数は

`bestChanges = 0`

で終了しており、これは変更が要求されないことを意味している。そうでない場合、変更を適用する必要がある。変更を適用する機能は、`targetNumber` との比較までは、（上述の擬似コードにおいて）変更をカウントすることと同一である。変更アプリケーションは以下の通りである：


```

if ((targetNumber < 1) || (targetNumber > TARGETS_PER_BLOCK))
{
    LocatedTargets[j] = NULL
    TargetsFound--
}
else
{
    LocatedTargets[j]->TargetNumber = targetNumber
}

```

10

変更ループの最後に、LocatedTargetsリストをコンパクトにして、全てのヌルターゲットを削除する必要がある。

【0587】

この手続の最後に、数個のターゲットが存在する可能性がある。残っている全てのターゲット（少なくとも2個のターゲットが必要である）は、クロックマーク及びデータ領域の位置を検出するため使用される。

20

【0588】

図55に示されるように、上側領域の第1のクロックマークドット1126は、第1のTarget1124の中心から55ドット離れている（これはターゲット中心間の距離と同じである）。クロックマークドットの中心は更に1ドット遠ざかり、黒色境界線1123は第1のクロックマークドットから更に4ドット離れている。下側領域の第1のクロックマークドットは、上側領域の第1のクロックマークドット1126から正確に7ターゲットの距離（ 7×55 ドット）離れている。

【0589】

Target1及び6が見つっていると仮定し得ないので、最も上側のターゲット及び最も下側のターゲットを使用する必要がある、どのターゲットが使用されているかを判定するためターゲット番号を使用する。この時点で少なくとも2個のターゲットが必要である。その上、ターゲット中心は、実際のターゲット中心の推定値に過ぎない。それは、ターゲット中心をより正確に位置検出することである。ターゲットの中心は、黒色に囲まれた白色である。したがって、画素次元と列次元の両方で極大を見つけることが望ましい。これには、連続した画像を再構築することが必要である。なぜならば、最大値が整数バウンダリ（本例の推定値）に厳密に合わせられる可能性は殆ど無いからである。

30

【0590】

連続した画像をターゲットの中心の周りに構築する前に、2個のターゲット中心のより良い推定値を作成することが必要である。既存のターゲット中心は、実施には、ターゲット中心の境界ボックスの左上座標である。ターゲットの中心を画成するエリアを見つけるため各画素を一つずつ調べ、最高値を有する画素を検出することは、簡単なプロセスである。同じ最高画素値を有する画素は2個以上存在する場合もあるが、中心値の推定には1個の画素があればよい。

40

【0591】

擬似コードは簡単であり、2個のターゲットの各々に対して実行される：

CENTER_WIDTH = CENTER_HEIGHT = 12

maxPixel = 0x00

for (i=0; i<CENTER_WIDTH; i++)

for (j=0; j<CENTER_HEIGHT; j++)

{

p = GetPixel(column+i, pixel+j)

if (p > maxPixel)

{

maxPixel = p

centerColumn = column + i

centerPixel = pixel + j

}

}

Target.Column = centerColumn

Target.Pixel = centerPixel

10

20

30

このプロセスの最後に、ターゲット中心座標は、実際の中心の1画素の範囲に収まるべきターゲットの最も白い画素を指示する。ターゲット中心のより正確な位置を構築するプロセスには、推定されたターゲット中心の両側に3個ずつで、ターゲットの7個の走査線スライスの連続信号を再構築することが含まれる。検出された7個の最大値（上記の画素スライスの各々につき1個ずつ）は、列次元において連続信号を再構築し、次に、その次元における最大値の位置を検出するために使用される：

40

```

// Given estimates column and pixel, determine a
// betterColumn and betterPixel as the center of
// the target
for (y=0; y<7; y++)
{
    for (x=0; x<7; x++)
        samples[x] = GetPixel(column-3+y, pixel-3+x)
    FindMax(samples, pos, maxVal)
    reSamples[y] = maxVal
    if (y == 3)
        betterPixel = pos + pixel
}
FindMax(reSamples, pos, maxVal)
betterColumn = pos + column

```

10

20

FindMaxは、元の1次元信号に基づくサンプル点を再構築し、最大値の位置及び検出された最大値を返す関数である。使用される信号再構築／再サンプリング方法は、図76に示されるように、Lanczos3窓sinc関数である。

【0592】

Lanczos3窓sinc関数は、再構築された次元から、推定位置Xの周りに集められた、即ち、 $X-3$ 、 $X-2$ 、 $X-1$ 、 X 、 $X+1$ 、 $X+2$ 、 $X+3$ にある7（画素）サンプルを取得する。 $X-1$ から $X+1$ の間に、0.1の間隔で点を再構築し、最大となる点を決定する。最大値である位置が新しい中心になる。カーネルの性質上、 X と $X+1$ の間の点に対してコンボリューションカーネルに必要なエントリは6個だけである。 $X-1$ から X に対して6個の点と使用し、 X から $X+1$ に対して6個の点を使用し、 $X-1$ から $X+1$ 間での画素値を得るために全体として7点が必要である。なぜならば、要求された画素の一部は一致するからである。

30

【0593】

最も上側のターゲットから最も下側のターゲットまでに正確な推定値が与えられた場合、上側領域及び下側領域に対して第1のクロックマークドットの位置を以下のようにして計算することができる：

```

TARGETS_PER_BLOCK = 6
numTargetsDiff = to.TargetNum - from.TargetNum
deltaPixel = (to.Pixel - from.Pixel) / numTargetsDiff
deltaColumn = (to.Column - from.Column) / numTargetsDiff
UpperClock.pixel = from.Pixel - (from.TargetNum * deltaPixel)
UpperClock.column = from.Column - (from.TargetNum * deltaColumn)

```

```

// Given the first dot of the upper clockmark, the
// first dot of the lower clockmark is straightforward.

```

10

```

LowerClock.pixel          =          UpperClock.pixel          +
      ((TARGETS_PER_BLOCK+1) * deltaPixel)
LowerClock.column         =          UpperClock.column         +
      ((TARGETS_PER_BLOCK+1) * deltaColumn)

```

これにより第1のクロックマークドットが得られる。列位置は、クロックマークの中心へ到達するため、データエリアから更に1ドット遠くへ動かすことが必要である。また、画素位置は、境界線の中心へ到達するため、更に4ドット遠ざける必要がある。deltaColumn及びdeltaPixelの疑似コード値は、55ドットの距離（ターゲット間の距離）に基づいているので、これらのデルタは、クロックマーク座標に適用される前に、それぞれ、 $1/55$ 及び $4/55$ によって縮小しなければならない。これは以下の通りに表現される：

20

```

kDeltaDotFactor =
1/DOTS_BETWEEN_TARGET_CENTRES
deltaColumn *= kDeltaDotFactor
deltaPixel *= 4 * kDeltaDotFactor
UpperClock.pixel -= deltaPixel
UpperClock.column -= deltaColumn
LowerClock.pixel += deltaPixel
LowerClock.column += deltaColumn

```

30

UpperClock及びLowerClockは、ターゲットの中心とちょうど合致した第1のクロックマークに対する有効クロックマーク推定値である。

【0594】

黒色及び白色画素／ドットレンジの設定

データをデータエリアから抽出する前に、黒色及び白色ドットの画素レンジを確認する必要がある。LookingForTargets中に現れた最小値及び最大値は、それぞれ、WhiteMin及びBlackMaxに保存されているが、これらは、データ抽出に関してこれらの変数の有効な値を表現していない。それらは、単に、記憶の便宜上、使用されるだけである。以下の疑似コードは、出現した最小画素及び最大画素に基づいてWhiteMin及びBlackMaxの良い値を獲得する方法を示している：

40

```

MinPixel = WhiteMin
MaxPixel = BlackMax
MidRange = (MinPixel + MaxPixel)/2
WhiteMin = MaxPixel - 105
BlackMax = MinPixel + 84
CurrentState = ExtractingBitImage

```

ExtractingBitImage状態は、データブロックが、ターゲットによって既に正確に位置検出された状態であり、ビットデータは、同時に1ドット列毎に抽出中であり、代替Artcardビットイメージに書き込まれている。データブロッククロックマーク／境界の追跡によって、回転とは無関係に正確なドット復元が行われ、セグメント境界は無視される。

50

データブロック全体（595個のデータ列と2個の方位列を併せて、48バイトずつの597列）が抽出されると、新しいセグメント境界が現在位置に基づいて次のデータブロックのために計算される。この状態は、LookingForTargets状態に変更される。

【0595】

所与のドット列の処理には以下の二つのタスクが必要である：

＊第1のタスクは、クロックマークを用いて特定のデータのドット列の位置を検出することである；

＊第2のタスクは、ドット列を追跡して、1ドットにつき1ビットずつビット値を集めることである。

【0596】

これらの二つのタスクは、列のデータが代替A r t c a r dから読み出され、D R A Mへ転送された場合に限り行うことができる。これは、プロセス1がどの走査線まで到達しているかをチェックし、それをクロックマーク列と比較することによって判定される。ドットデータがD R A Mに存在するとき、クロックマークを更新し、クロックマークを次のドット列に対する推定値へ進める前に、列からデータを抽出することが可能である。このプロセスの概要は、以下の疑似コードに示されている。具体的な関数については後述する：

```

finishedBlock = FALSE
if((UpperClock.column < Process1.CurrentScanLine)
    &&
    (LowerClock.column < Process1.CurrentScanLine))
{
    DetermineAccurateClockMarks()
    DetermineDataInfo()
    if (CurrentDotColumn >= 0)
        ExtractDataFromColumn()
    AdvanceClockMarks()
    if (CurrentDotColumn == FINAL_COLUMN)
    {
        finishedBlock = TRUE
        currentState = LookingForTargets
        SetBounds(UpperClock.pixel, LowerClock.pixel)
        BitImage += 256KB
        CurrentByte = 0
        TargetsFound = 0
    }
}
return finishedBlock

```

ドット列の位置検出

所定のドット列は、ドットを読み出し、データを抽出する前に、位置検出されなければな

10

20

30

40

50

らない。これは、データブロックの上側及び下側境界に沿ってクロックマーク／境界線を追跡することによって実現される。位相ロックループと等価的なソフトウェアが使用され、たとえクロックマークが破損されていても、クロックマーク位置の良い推定が確実に行われる。図 77 は、例示的なデータブロックの左上の説明図であり、そのデータブロックのコーナーには、ターゲットエリアの外側へ広がる 3 ドット高のクロックマーク 1 1 6 6 と、白色行と、黒色境界線と、が現されている。

【0597】

初期的に、第 1 の黒色クロックマークの位置の推定値が（ターゲット位置に基づいて）提供される。正確な垂直位置（画素）を得るため黒色境界が使用され、正確な水平位置（列）を得るため、例えば、クロックマーク 1 1 6 6 が使用される。これらは、UpperClock 及び LowerClock の位置に反映される。

10

【0598】

クロックマーク推定値が取得され、その近傍の画素を調べることにより、連続した信号が再構築され、正確な中心が決定される。2 次元をクロックマークと境界に分解しているので、これは、簡単な 1 次元プロセスであり、2 回実行する必要がある。しかし、これは、位置合わせの相手側の黒色クロックマークが存在するとき、2 ドット列について 1 回だけ行われる。白色クロックマークの場合、単に推定値を使用し、推定値はそのままにしておく。或いは、（常に存在するので）ドット列毎に境界に基づいて画素座標を更新してもよい。實際上、（黒色クロックマークを含む）1 列おきに両方の座標を更新すれば十分である。なぜならば、この動作中の解像度は非常に精細だからである。したがって、このプロセスは次のようになる：

20

// Turn the estimates of the clockmarks into accurate

// positions only when there is a black clockmark

// (ie every 2nd dot column, starting from -8)

if (Bit0(CurrentDotColumn) == 0) // even column

{

DetermineAccurateUpperDotCenter()

DetermineAccurateLowerDotCenter()

}

30

所定の許容度（MAX_CLOCKMARK_DEVIATION）を上回る偏差が存在する場合、検出された信号は無視され、推定値からの最大許容度に収まる偏差だけが許容される。この点に関して、この機能はフェーズロックループの機能と類似している。したがって、DetermineAccurateUpperDotCenter 関数は以下の疑似コードによって実現される：

```

// Use the estimated pixel position of
// the border to determine where to look for
// a more accurate clockmark center. The clockmark
// is 3 dots high so even if the estimated position
// of the border is wrong, it won't affect the
// fixing of the clockmark position.
MAX_CLOCKMARK_DEVIATION = 0.5
diff = GetAccurateColumn(UpperClock.column,
                          UpperClock.pixel+(3*PIXELS_PER_DOT))
diff -= UpperClock.column
if (diff > MAX_CLOCKMARK_DEVIATION)
    diff = MAX_CLOCKMARK_DEVIATION
else
    if (diff < -MAX_CLOCKMARK_DEVIATION)
        diff = -MAX_CLOCKMARK_DEVIATION
    UpperClock.column += diff

```

10

```

// Use the newly obtained clockmark center to
// determine a more accurate border position.
diff = GetAccuratePixel(UpperClock.column, UpperClock.pixel)
diff -= UpperClock.pixel
if (diff > MAX_CLOCKMARK_DEVIATION)
    diff = MAX_CLOCKMARK_DEVIATION
else
    if (diff < -MAX_CLOCKMARK_DEVIATION)
        diff = -MAX_CLOCKMARK_DEVIATION
    UpperClock.pixel += diff

```

20

30

また、DetermineAccurateLowerDotCenter関数は、境界からクロックマークまでの向きが負方向である（＋３ドットではなく、－３ドットである）点を除くと同様である。

【 0 5 9 9 】

GetAccuratePixel及びGetAccurateColumnは、座標が与えられた場合に、一次元だけの観点で正確なドット中心を決定する関数である。正確なドット中心の決定は、信号を再構成し、最小信号値が検出された位置を見つけるプロセスである（これは、ターゲット中心は黒色ではなく白色であるために、信号の最大値の位置を検出することによりターゲット中心の位置を検出するプロセスとは異なる）。このアプリケーションの信号再構成／再サンプリングのために選択された方法は、図 7 6 を参照して説明したようなLanczos3窓sinc関数である。

40

【 0 6 0 0 】

クロックマーク又は境界は何らかの方法で、おそらくは、ひっかきによって、傷つけられている可能性がある。再サンプリングによって得られた新しい中心値と推定値との差が許容量を超える場合、中心値は最大許容量だけ動かされる。それが無効な位置である場合、データ回復に使用するために十分に接近しているはずであり、更なるクロックマークはその位置に再度合わせられる。

【 0 6 0 1 】

第 1 のデータドットの中心と後続のドットへのデルタの決定

50

正確なUpperClock及びLowerClock位置が決定されると、第1のデータドットの中心（CurrentDot）と、列内の後続のドットへ進めるためにその中心位置に加算されるべきデルタ量（DataDelta）とを計算することができる。

【0602】

最初に行うべきことは、ドット列のデルタを計算することである。これは、単に、LowerClockからUpperClockを減算し、2点間のドット数で除算することによって実現される。実際にはドット数の逆数を乗算することが可能である。なぜならば、ドット数は代替Artcardに関して定数であり、且つ乗算の方が高速に行われるからである。画素次元のデルタと列次元のデルタを獲得するため、異なる定数を使用することが可能である。画素のデルタは二つの境界間の距離であり、列のデルタは二つのクロックマークの中心間である。このようにして、関数DetermineDataInfoは二つの部分からなる。第1の部分は次の擬似コードによって表される：

```
kDeltaColumnFactor =
1/(DOTS_PER_DATA_COLUMN+2+2-1)
```

```
kDeltaPixelFactor =
1/(DOTS_PER_DATA_COLUMN+5+5-1)
```

```
delta = LowerClock.column -
UpperClock.column
```

```
DataDelta.column = delta *
```

```
kDeltaColumnFactor
```

```
delta = LowerClock.pixel - UpperClock.pixel
```

```
DataDelta.pixel = delta * kDeltaPixelFactor
```

次に、列の第1のデータの中心を決定することが可能である。クロックマークの中心から第1のデータドットの中心までの距離は2ドットであり、境界の中心から第1のデータドットの中心までの距離は5ドットである。この関数の第2の部分は次の擬似コードによって表される：

```
CurrentDot.column = UpperClock.column + (2*DataDelta.column)
```

```
CurrentDot.pixel = UpperClock.pixel +
(5*DataDelta.pixel)
```

ドット列の追跡

ドット列はクロックマークに追従する位相ロックループから見つけられたので、残っていることは、その列を下って各ドットの中心でドット列をサンプリングすることだけである。変数CurrentDot点は現在列の第1のドットの中心に決定される。列の次のドットへ進むためには、DataDeltaを加算するだけでよい（2回の加算：1回は列座標のため、もう1回は画素座標のため）。所定の座標におけるドットのサンプリング（バイリニア補間）が行われ、ドットの中心を表現する画素値が決定される。画素値は、次に、そのドットのビット値を決定するため使用される。しかし、より良いビット判定を行うため、同一ドットライン上の2個の周辺ドットに対する中心値と関連して、その画素値を使用することが可能である。

【0603】

抽出されたドット列内のドットに対する全ての画素はこの時点でDRAMにロードされていると仮定してもよい。なぜならば、ラインの両端（クロックマーク）がDRAMに存在するならば、これらの2個のクロックマークの間のドットも間違いなくDRAMに存在するからである。更に、データブロック高さは十分に低いので（384ドット高にすぎない）、単なるデルタだけがラインの全長を越えるために十分であることが保証される。カードが8個のデータブロック高さに分割されている理由の一つは、単一のデータブロックに関して行うことができる保証と同じ厳格な保証を、カードの全高に亘って行うことが不可能だからである。

【0604】

単一ラインのデータ（48バイト）を抽出する上位レベルプロセスは、以下の擬似コード

で表される。データバッファポインタは、各バイトが保存されるのに伴ってインクリメントされ、連続したバイト及びデータの列が確実に連続的に保存される：

```

bitCount = 8
curr = 0x00                                // definitely black
next = GetPixel(CurrentDot)
for (i=0; i < DOTS_PER_DATA_COLUMN; i++)
{
    CurrentDot += DataDelta
    prev = curr
    curr = next
    next = GetPixel(CurrentDot)
    bit = DetermineCenterDot(prev, curr, next)
    byte = (byte << 1) | bit
    bitCount--
    if (bitCount == 0)
    {
        *(BitImage | CurrentByte) = byte
        CurrentByte++
        bitCount = 8
    }
}

```

10
20
30

GetPixel関数はドット座標（固定小数点）を取得し、バイリニア補間によって中心画素値に達するため、4個のCCD画素をサンプリングする。

【0605】

DetermineCenterDot関数は、ビット値が決定されたドットの両側にあるドット中心を表現する画素値を取得し、その中心ドットのビット値の値を知的に推定することを試みる。図64の一般化されたブラー効果曲線から、以下の三つの一般的なケースが考えられる：*ドットの中心画素値はWhiteMinよりも小さいので、確実に黒色ドットである。したがって、ビット値は必ず1である；

40

*ドットの中心値はBlackMaxよりも大きいので、確実に白色ドットである。したがって、ビット値は必ず0である；

*ドットの中心値は、BlackMaxとWhiteMinの間のどこかにある。ドットは黒色かもしれないし、白色かもしれない。したがって、このビットの値が問題になる。ビットの値を合理的に推定するため、多数の方式が考えられている。これらの方式は、精度に対して複雑さのバランスを取り、ある種のケースでは、保証された解決策が存在しないことを考慮する必要がある。誤ったビット判定を行うケースでは、ビットのリード・ソロモンシンボルに誤りが生じ、フェーズ2のリード・ソロモン復号化ステージで訂正しなければならない。

【0606】

画素値がBlackMaxとWhiteMinの間に入るときにドットの値を決定するため使用される方式

50

は、あまり複雑ではないが、優れた結果を与える。この方式は、門対となっているドットの左及び右にあるドット中心の画素値を使用して、中心ドットのよりもらしい値を決定する。即ち、

＊両側の２個のドットがMidRange（平均ドット値）の白色側に存在する場合、中心ドットが白色であるならば、「確実に」白色であると推定することができる。その中心ドットが不確実な領域に存在するということは、そのドットが黒色であり、その値をより不確実にするため周辺の白色ドットによる影響を受けたということを示す。したがって、そのドット値は黒色であると考えられるので、そのビット値は１である；

＊両側の２個のドットがMidRangeの黒色側に存在する場合、中心ドットが黒色であるならば、「確実に」黒色であると推定することができる。その中心ドットが不確実な領域に存在するということは、そのドットが白色であり、その値をより不確実にするため周辺の黒色ドットによる影響を受けたということを示す。したがって、そのドット値は白色であると考えられるので、そのビット値は０である；

＊一方のドットがMidRangeの黒色側に存在し、もう一方のドットがMidRangeの白色側に存在するならば、中心ドット値をそのまま使用して決定する。中心ドットがMidRangeの黒色側に存在するならば、黒色（ビット値１）を選択する。さもなければ、白色（ビット値０）を選択する。

【０６０７】

このロジックは以下のように表現される：

```

if (pixel < WhiteMin)                                // definitely black
    bit = 0x01
else
    if (pixel > BlackMax)                             // definitely white
        bit = 0x00
    else
        if ((prev > MidRange) && (next > MidRange)) //prob black
            bit = 0x01
        else
            if ((prev < MidRange) && (next < MidRange)) //prob white
                bit = 0x00
            else
                if (pixel < MidRange)
                    bit = 0x01
                else
                    bit = 0x00

```

ここから、周辺画素値の使用は、中心ドットの状態の優れた指標を与え得ることが認められる。ここで説明した方式は、同じ行からのドットだけを使用しているが、単一のドットライン履歴（前のドットライン）の使用は、明らかに代替的な構成になるであろう。

【０６０８】

次の列のためのクロックマーク更新

ある列の第1のデータドットの中心が決定されると、クロックマーク値は不要になる。それらは、その列に対するデータが獲得された後、次の列の準備のため更新する方が好都合である。クロックマークの向きは、ドット列に沿って下りるドットの追跡方向に対して垂直であるため、列を更新するため画素デルタを使用し、両方のクロックの画素を更新するため列デルタを使用することが可能である：

```
UpperClock.column += DataDelta.pixel
```

```
LowerClock.column += DataDelta.pixel
```

```
UpperClock.pixel -= DataDelta.column
```

```
LowerClock.pixel -= DataDelta.column
```

この時点で、次のドット列の推定値が得られる。

10

【0609】

タイミング

タイミングの必要条件は、DRAM利用率が100%を超えない限り満たされ、並列アルゴリズムの追加によってタイミングが増大されるがDRAM利用率は100%を超えない。DRAM利用率はプロセス1に関して特定され、このプロセス1は各画素を1回ずつ連続的に書き込み、DRAM帯域幅の9%を消費する。

【0610】

このセクションに記載されるようなタイミングは、DRAMが代替Art cardリーダーアルゴリズムの要求に容易に対処し得ることを示す。したがって、タイミングのボトルネックは、DRAMアクセスではなく、ロジック速度に関するアルゴリズムの実装であろう。しかし、アルゴリズムは、アーキテクチャの単純さに留意して設計され、メモリサイクル毎に最小限の個数の論理演算しか要求しない。この点で、実装された状態機械又は等価的なCPU/DSPアーキテクチャが以下のサブセクションに記載されているように実行可能である限り、目標速度は達成されるであろう。

20

【0611】

ターゲット位置検出

ターゲットは画素列の境界内の画素を読み出すことによって位置検出される。各画素は高々1個ずつ読み出される。ランレングスエンコーダが十分高速に動作する場合を考えると、ターゲットの位置検出の限界はメモリアクセスである。したがって、アクセスはプロセス1のタイミング、即ち、DRAM帯域幅の9%利用率よりは悪化しないであろう。

30

【0612】

したがって、ターゲット位置検出中の（プロセス1を含む）DRAMの総利用率は18%であり、ターゲットロケータは、常に、代替Art cardのイメージセンサ画素リーダーに追従している。

【0613】

ターゲットの処理

ターゲット番号をソートしチェックするためのタイミングは取るに足らない。2個のターゲット中心の各々のより良い推定値を見つけるためには、12組の12画素の読み出し、即ち、全部で144回の読み出しを必要とする。しかし、正確なターゲット中心の修復は些細ではなく、2組の推定を必要とする。各ターゲット中心の調整には、8組の20種類の6エン트리型コンボリョーションカーネルが必要である。このようにして、その合計は、 $8 \times 20 \times 6$ 回の乗算－累算＝960に達する。その上、7組の7画素を獲得するため49回のメモリアクセスが必要である。したがって、1ターゲット当たりの総数は、 $144 + 960 + 49 = 1153$ であり、これは、画素の列内の画素数（1152）とほぼ同数である。このように、各ターゲット推定は、1行の画素を処理するために要する時間を消費する。ターゲットが2個の場合、実質的に2列の画素列に要する時間を消費する。

40

【0614】

ターゲットは、第1の画素列上でターゲット番号の後に確実に識別される。方位列の前には2列のドット列が存在するので、6列の画素列が存在する。ターゲット位置検出プロセスは、最初の画素列を使い果たすが、残りの5列の画素列は全く処理されない。したがっ

50

て、データエリアは、他のプロセス時間に影響を与えることなく、利用可能な時間の2/5内に位置検出することができる。

【0615】

利用可能な時間の残りの3/5は、白色及び黒色画素のレンジを割り当てる些細なタスクのために十分であり、1タスクは高々2から3マシンサイクルを使用する。

【0616】

データ抽出

タイミングに関して、次の二つの部分を考慮すべきである：

*正確なクロックマーク及び境界値を取得すること；

*ドット値を抽出すること。

10

【0617】

クロックマーク及び境界値は、1ドット列おきにしか収集されない。しかし、より正確にするためにクロックマーク推定値を更新する毎に、20種類の6エンタリー型コンボリューションカーネルを評価しなければならない。平均的に、1ドット列当たりにはそのうちの2個が存在する（2ドット列毎に4個）。境界に基づいて画素座標を更新するためには、同じ画素走査線からの7画素だけが必要である。しかし、列座標を更新するためには、異なる列、即ち、異なる走査線からの7画素が必要である。走査線エンタリー毎に1回のキャッシュミスが生じ、同じ走査線内の画素に対して2回のキャッシュミスが生じる最悪ケースの状況を想定すると、全部で8回のキャッシュミスが起こる。

【0618】

20

ドット情報を抽出するためには、1ドット毎に（ドットを定義する平均9個ではなく）4画素だけを読み出す。1152画素（384ドット）のデータエリアを想定すると、せいぜい、これは、9画素ドットではなく4画素ドットを読み出すことによって、72回のキャッシュ読み出しを節約するに過ぎない。最悪ケースは、57画素毎に1列だけ移動する1°の回転であるが、節約の程度が僅かに悪化するだけである。

【0619】

最悪の場合、読み出すキャッシュラインの数は、データエリア内の画素によって使われたキャッシュラインよりも少なくなる、といっても構わない。したがって、アクセスはプロセス1、即ち、DRAM帯域幅の9%利用率よりも悪化することはない。

【0620】

30

したがって、データ抽出中（プロセス1を含む）のDRAM総利用率は18%であり、データ抽出部は、常に、代替Artcardのイメージセンサ画素リーダーに追従する。これにより実装されるターゲット処理プロセスでは、ターゲットの処理は、かなり非効率的な方法によって実行されるが、依然として、必要に応じて、データ抽出プロセス中に迅速に追従する。

【0621】

フェーズ2ービットイメージ復号

フェーズ2は、代替Artcardのデータ復元アルゴリズムの非リアルタイムフェーズである。フェーズ2の先頭で、ビットイメージは代替Artcardから抽出されている。ビットイメージは、代替Artcardのデータ領域から読み出されたビットを表現する。一部のビットには誤りがあり、おそらく、代替Artcardは挿入時に回転させられているので、データ全体は180°回転している。フェーズ2は、この符号化されたビットイメージから原データを確実に抽出することに関係している。基本的に、図79に示されるように3ステップが実行される：

40

*代替Artcardが逆向きに挿入されているならば反転して、ビットイメージを再編成する；

*符号化されたデータをスクランブル解除する；

*ビットイメージからのデータをリード・ソロモン復号化する。

【0622】

3ステップの各々は別個のプロセスとして定義され、あるプロセスの出力は次のプロセル

50

の入力として要求されるので、連続して実行される。最初の 2 ステップを単一プロセスに併合することは簡単であるが、説明の便宜上、ここでは、別個のプロセスとして取り扱う。

【0623】

データ／プロセスの観点からみると、フェーズ 2 は図 80 に示されるような構造を有する。

【0624】

プロセス 1 及び 2 のタイミングは無視することが可能であり、その間に 1 / 1000 秒未満しか消費しない。プロセス 3 (リード・ソロモン復号) は、約 0.32 秒を消費し、これがフェーズ 2 のために必要な合計時間になる。

10

【0625】

ビットイメージ再編成、必要に応じて反転

DRAM 内のビットマップは、この時点で、代替 Art card から取得されたデータを表現する。しかし、ビットイメージは連続していない。ビットイメージは 64 個の 32 k 単位のチャンクに分割され、データブロック毎に 1 チャンクがある。32 k チャンクの各々は、28656 個の有効バイトだけ、即ち、

最も左側の方位列からの 48 バイトと、

真のデータ領域からの 28560 バイトと、

最も右側の方位列からの 48 バイトと、

未使用の 4112 バイトと、

20

を格納する。

【0626】

画素データ (フェーズ 1 のプロセス 1 によって格納) のために使用される 2 MB バッファは、再編成されたビットイメージを保持するために使用可能である。なぜならば、画素データはフェーズ 2 の間では要求されないからである。再編成の最後に、正しく向きを与えられた連続したビットイメージは、2 MB 画素バッファに収容され、リード・ソロモン復号化のための準備ができています。

【0627】

カードの向きが正しい場合、最も左側の方位列は白色であり、最も右側の方位列は黒色である。カードが 180° 回転している場合、最も左側の方位列は黒色であり、最も右側の方位列は白色である。

30

【0628】

カードが正しい向きにされているかどうかを判定する簡単な方法は、白色ビットに対する黒色ビットの比率が圧倒的であるブロックが見つかるまで、各データブロックを一つずつ使用して、データの最初及び最後の 48 バイトをチェックすることである。以下の擬似コードはこの方法を例示し、カードが正しい向きに置かれているとき TRUE を返し、そうではなければ FALSE を返す：

```

totalCountL = 0
totalCountR = 0
for (i=0; i<64; i++)
{
    blackCountL = 0
blackCountR = 0
currBuff = dataBuffer
for (j=0; j<48; j++)
{
    blackCountL += CountBits(*currBuff)
    currBuff++
}
currBuff += 28560
for (j=0; j<48; j++)
{
    blackCountR += CountBits(*currBuff)
    currBuff++
}
dataBuffer += 32k
if (blackCountR > (blackCountL * 4))
    return TRUE
    if (blackCountL > (blackCountR * 4))
        return FALSE
    totalCountL += blackCountL
    totalCountR += blackCountR
}
return (totalCountR > totalCountL)

```

データは、この時点で、カードが正しい向きであるかどうかに基づいて認識できるはずである。最も簡単なケースは、カードが正しい向きにされている場合である。この場合、データは、方位列を除去し、データ全体を連続させるために、少しだけ動かす必要がある。これは、以下の擬似コードに示されるように元の位置で非常に簡単に実現される：

```

DATA_BYTES_PER_DATA_BLOCK = 28560
to = dataBuffer
from = dataBuffer + 48)           // left orientation column
for (i=0; i<64; i++)
{
    BlockMove(from, to, DATA_BYTES_PER_DATA_BLOCK)
    from += 32k
    to += DATA_BYTES_PER_DATA_BLOCK
}

```

10

それ以外の場合は、データを実際に反転させる必要がある。データを反転させるアルゴリズムは非常に簡単であるが、簡単にするために、256バイトのテーブルReverseを用いる。ここで、Reverse[N]の値はNのビット反転である：

```

DATA_BYTES_PER_DATA_BLOCK = 28560
to = outBuffer
for (i=0; i<64; i++)
{
    from = dataBuffer + (i * 32k)
    from += 48                     // skip orientation column
    from += DATA_BYTES_PER_DATA_BLOCK - 1 // end of block
    for (j=0; j < DATA_BYTES_PER_DATA_BLOCK; j++)
    {
        *to++ = Reverse[*from]
        from--
    }
}

```

20

30

両方のプロセスのタイミングは無視することが可能であり、1 / 1000 秒未満しか消費しない： 40

* 2 MB 連続読み出し ($2048 / 16 \times 12 \text{ ns} = 1536 \text{ ns}$) ;

* 2 MB 実効的連続バイト書き込み ($2048 / 16 \times 12 \text{ ns} = 1536 \text{ ns}$) 。

【0629】

符号化画像のスクランブル解除

ビットイメージは、1827840個の連続した、正しい向きにされた、スクランブル付きのバイトである。バイトは、1ブロックが255バイト長からなる7168個のリード・ソロモンブロックを生成するために、スクランブル解除されなければならない。スクランブル解除プロセスは非常に簡単であるが、スクランブル解除処理は元の場所で行えないので、別個の出力バッファを必要とする。図80は、スクランブル解除プロセスが行

50

われるメモリの説明図である。

【0630】

以下の疑似コードはスクランブル解除プロセスの実行方法を規定する：

groupSize = 255

numBytes = 1827840;

inBuffer = scrambledBuffer;

10

outBuffer = unscrambledBuffer;

for (i=0; i<groupSize; i++)

for (j=i; j<numBytes; j+=groupSize)

20

outBuffer[j] = *inBuffer++

このプロセスのためのタイミングは無視することが可能であり、1 / 1000 秒未満しか消費しない：

＊2MB連続読み出し（ $2048 / 16 \times 12 \text{ ns} = 1536 \text{ ns}$ ）；

＊2MB非連続バイト書き込み（ $2048 \times 12 \text{ ns} = 24576 \text{ ns}$ ）。

【0631】

このプロセスの終了時、スクランブル無しデータがリード・ソロモン復号化のために準備できる。

30

【0632】

リード・ソロモン復号化

代替ArtCardの読み出しの最後の部分は、リード・ソロモン復号化プロセスであり、ここでは、約2MBのスクランブル無しデータが約1MBの有効代替ArtCardデータに復号化される。

【0633】

このアルゴリズムは、同時に一つのリード・ソロモンブロックを復号化し、（必要であれば）元の場所で実行可能である。なぜならば、符号化されたブロックは復号化されたブロックよりも大きく、冗長バイトがデータバイトの後に格納されているからである。

40

【0634】

最初の2個のリード・ソロモンブロックは制御ブロックであり、ビットイメージから抽出されるべきデータのサイズに関する情報を含んでいる。このメタ情報は、最初に復号化する必要がある、得られた情報が真のデータを復号化するため使用される。真のデータの復号化は、単に同時に一つずつデータブロックを復号化する問題である。特定のブロックの復号化が失敗した場合、控えのデータブロックを使用することが可能である。

【0635】

最上位レベルのリード・ソロモン復号化を疑似コードで提示する：


```

// Constants for Reed Solomon decode
sourceBlockLength = 255;
destBlockLength = 127;
numControlBlocks = 2;

// Decode the control information
if (! GetControlData(source, destBlocks, lastBlock))
    return error

destBytes = ((destBlocks-1) * destBlockLength) + lastBlock
offsetToNextDuplicate = destBlocks * sourceBlockLength

// Skip the control blocks and position at data
source += numControlBlocks * sourceBlockLength

// Decode each of the data blocks, trying
// duplicates as necessary
blocksInError = 0;
for (i=0; i<destBlocks; i++)
{
    found = DecodeBlock(source, dest);
    if (! found)
    {
        duplicate = source + offsetToNextDuplicate
        while ((! found) && (duplicate<sourceEnd))
        {
            found = DecodeBlock(duplicate, dest)
            duplicate += offsetToNextDuplicate
        }
        if (! found)
            blocksInError++

        source += sourceBlockLength
        dest += destBlockLength
    }
}

return destBytes and blocksInError

```

10

20

30

40

DecodeBlockは、標準的なリード・ソロモンブロックデコーダであり、 $m = 8$ 及び $t = 6$ 4を使用する。

【 0 6 3 6 】

GetControlData関数は、復号化エラーが生じない限り簡単である。この関数は、成功するまで、同時に一つの制御ブロックを復号化するためDecodeBlockを呼び出すだけである。制御パラメータは、復号化されたデータの最初の3バイトから抽出することができる (destBlocksはバイト0及び1に格納され、lastBlockはバイト2に格納される)。復号化エラーが存在する場合、この関数は、32組の3バイトをトラバースし、最も確からしい値の組を決定しなければならない。一つの簡単な方法

50

は、3 バイトの中から2 個の連続した等しいコピーを見つけ、これらの値が正しい値であるとして決めることである。他の方法は、出現する3 バイトの組の種類数をカウントし、最もよく出現する組が正しい組であると決める。

【0637】

リード・ソロモン復号化に要する時間は実装に依存する。リード・ソロモン復号化プロセスを実行するために専用コア（例えば、LSI ロジックのL64712）を使用可能であるが、アプリケーションに応じて（通常、復号化されたデータを用いて何かを実行するために）内蔵型システムの全体でより汎用的に使用できるCPU/DSPの組み合わせを選択する方が好ましい。勿論、復号化時間はCPU/DSPの組み合わせの場合にも十分に高速でなければならない。

10

【0638】

L64712は、毎秒50Mビット（毎秒約6.25MB）のスループットがあるので、時間は、最大で2MB読み出し及び1MB書き込みのメモリアクセス時間ではなく、リード・ソロモンデコーダの速度によって制約される。最悪のケース（全ての2MBが復号化を必要とするケース）に要する時間は、従って、2/6.25秒=約0.32秒である。勿論、以下の改良を含む多数の改良を考慮することができる。

【0639】

読み出し環境のブラー効果が大きいほど、所定のドットは周辺ドットからより大きい影響を受ける。好適な実施例による現在の読み出しアルゴリズムは、ドット値に関してより優れた決定を行うため、同一列内の周辺ドットを使用する能力を備えている。前の列のドットは既に復号化されているので、前の列のドット履歴は、画素値が不確実なレンジに含まれるドットの値を決定する際に役に立つ。

20

【0640】

初期ステージに関する別の可能性は、それを完全に排除し、データブロックの初期境界を必要とされる境界よりも広くし、ターゲット処理関数をより知的にすることである。これは、全体的な複雑さを軽減する。データブロックの独立性を維持するために注意する必要がある。

【0641】

制御ブロック機構はより頑強にすることができる：

＊制御ブロックは、（本例のケースのように）隣接させるのではなく、最初のブロック及び最後のブロックでもよい。これは、ある種の異常なダメージ状況に対する保護を高めるであろう；

30

＊第2の改良は、リード・ソロモン復号化ステップが失敗した場合に使用される補助的なレベルの冗長性／誤り検出を制御ブロック構造に取り入れることである。パリティのような簡単な仕組みは、リード・ソロモンステージが失敗した場合に、制御情報のもっともらしさを改良する。

【0642】

フェーズ5 Varkスクリプト実行

Artcard9を読み出し、それを復号化するために要する総時間は、したがって、約2.15秒である。ユーザがはっきりとわかる遅延は、実際には0.65秒（フェーズ3及び4の合計）に過ぎない。なぜならば、Artcardは1.5秒後に移動を停止するからである。

40

【0643】

Artcardがロードされると、Artvarkスクリプトを解釈しなければならない。スクリプトを直ちに実行するのではなく、スクリプトは「プリント」ボタン13（図1）が押下されたときだけに実行される。スクリプトを実行するための所要時間は、スクリプトの複雑さに応じて変化し、プリントボタンの押下と、実際のプリントボタンと、実際の印刷動作との間で認識される遅延を考慮しなければならない。

【0644】

上述の通り、VLIWプロセッサ74は、コンピュータ利用の費用がかかるVark関数

50

を加速するデジタル処理システムである。CPUコア72によってソフトウェアで実行される関数と、VLIWプロセッサ74によってハードウェアで実行される関数のバランスは、実装に依存する。VLIWプロセッサ74の目的は、全てのArt cardスタイルがユーザからみて遅いと感じられない時間内で実行することを助けることである。CPUが高速で強力になると共に、ハードウェア加速を必要とする関数の数は減少し続ける。VLIWプロセッサは、以下の時間的に重要な関数の汎用ハードウェア高速化を可能にするマイクロコード化されたALUサブシステムを含む：

- 1) 汎用ソフトウェア処理のための画像アクセス機構
- 2) 画像コンボルバ
- 3) データ駆動型画像ワーパー
- 4) 画像スケーリング
- 5) 画像モザイク
- 6) アフィン変換
- 7) 画像コンポジッター
- 8) 色空間変換
- 9) ヒストグラムコレクタ
- 10) 画像の照明
- 11) ブラシスタンパー
- 12) ヒストグラムコレクタ
- 13) CCDイメージから内部画像への変換
- 14) 画像ピラミッドの構築（ワーパーによってブラッシングのため使用される）。

10

20

【0645】

以下の表は、ALUモデルに実装された場合の各Vark演算の所要時間を要約したものである。ALUモデルを使用する関数を実装する方法は後述される。

【0646】

【表 5 7】

演 算	演算速度	1500*1000 画像	
		1 チャンネル	3 チャンネル
画像コンポジット	1 出力画素当たり 1 サイクル	0.015s	0.045s
画像コンボルブ	1 出力画素当たり k/3 サイクル (k = カーネルサイズ)		
	3×3 コンボルブ	0.045s	0.135s
	5×5 コンボルブ	0.125s	0.375s
	7×7 コンボルブ	0.245s	0.735s
画像ワープ	1 画素当たり 8 サイクル	0.120s	0.360s
ヒストグラムコレクト	1 画素当たり 2 サイクル	0.030s	0.090s
画像モザイク処理	1 画素当たり 1/3 サイクル	0.005s	0.015s
画像サブピクセル 平行移動	1 出力画素当たり 1 サイクル	—	—
色ルックアップ置換	1 画素当たり 1/2 サイクル	0.008s	0.023
色空間変換	1 画素当たり 8 サイクル	0.120s	0.360s
CCD 画像を内部画像 に変換(色変換及びス ケールを含む)	1 出力画素当たり 4 サイクル	0.06s	0.18s
画像ピラミッド構築	1 入力画素当たり 1 サイクル	0.015s	0.045s
スケール	1 入力画素当たり 2 サイクル	0.015s (最小値)	0.045s (最小値)
	1 出力画素当たり 2 サイクル		
	1 出力画素当たり 2 サイクル (X だ けでスケーリング) の最大値		
アフィン変換	1 出力画素当たり 2 サイクル	0.03s	0.09s
ブラシ回転／変換及び コンポジット	?		
タイル画像	1 出力画素当たり 4 から 8 サイクル	0.015 から 0.030s	4 チャンネルに対し 0.060 から 0.120s (Lab、テクスチャ)
画像照明	1 画素当たりサイクル数		
周囲のみ	1/2	0.008s	0.023s
指向性光	1	0.015s	0.045s
指向性 (bm)	6	0.09s	0.27s
オムニ光	6	0.09s	0.27s
オムニ (bm)	9	0.137s	0.41s
スポット光	9	0.137s	0.41s
スポット光 (bm)	12	0.18s	0.54s
(bm) = バンプマップ			

例えば、CCD 画像を変換するため、ヒストグラム収集、及び（画像強調用）ルックアップ色置換のために 1 画素当たり 9 + 2 + 0.5 サイクル、即ち、11.5 サイクルを要する。1500 × 1000 画像の場合、これは、1725000 サイクル、即ち、1 コンポーネント当たり約 0.2 秒、即ち、全 3 コンポーネント当たり 0.6 秒である。簡単なワープを追加すると、全体で、0.6 + 0.36、即ち、ほぼ 1 秒になる。

【0647】

画像コンボルブ

コンボルブは中心画素周りの重み付き平均化である。この平均は、単純な加算でもよく、絶対値の和でもよく、和の絶対値でもよく、或いは、0 で丸められた和でもよい。

【0648】

画像コンボルブは汎用コンボルブであり、可変サイズ係数カーネル内の値を変えることに

よって様々な関数を実現できる。サポートされるカーネルサイズは、 3×3 、 5×5 、及び 7×7 だけである。

【0649】

図82を参照すると、コンボリユーシヨンプロセスの一例340が示されている。コンボルバプロセス341に供給される画素コンポーネント値は、ボックス読み出しイタレータ342から与えられる。イタレータ342は行単位で画像データを供給し、各行内では、画素毎に画像データを供給する。コンボルバ341からの出力は、順次書き込みイタレータ344へ送られ、順次書き込みイタレータ344は有効画像フォーマットで合成画像を格納する。

【0650】

係数カーネル346は、DRAM内のルックアップテーブルである。カーネルは、ボックス読み出しイタレータ342と同じ順序に係数が並べられる。各係数エントリーは8ビットである。単純な順次読み出しイタレータは、カーネル346を指定し、係数を与えるため使用される。これは、カーネルサイズと一致する画像幅ImageWidthの画像をシミュレーションし、ループオプションがセットされるので、カーネルは連続して供給される。

【0651】

ALUユニット上のコンボルバプロセスの一実施形態は図81に示されている。以下の定数がソフトウェアによってセットされる。

【0652】

【表58】

定数	値
K_1	カーネルサイズ(9,25,又は49)

制御ロジックは、1画素当たりの乗算／加算の回数をカウントダウンするため使用される。カウント（ラッチ2に累算）が0に達するとき、生成される制御信号は（ラッチ1からの）現在のコンボルバ値を書き出し、カウントをリセットするため使用される。このようにして、一つの制御ロジックブロックは、多数の並列コンボルバストリームのために使用される。

【0653】

各サイクルに、乗算ALUは、画素の適切な部分を組み込むため、1回の乗算／加算を実行することができる。全ての値を総計するために要するサイクル数は、したがって、カーネル内のエントリーの個数である。これは計算制約であるため、画像を複数のセクションに分割し、それらを異なるALUユニットで並列に処理することが適切である。

【0654】

7×7 カーネル上で、各画素に要する時間は49サイクル、即ち、 490 ns である。各キャッシュラインは32画素を保持するので、メモリアクセスのために利用可能な時間は、 $((32 - 7 + 1) \times 490 \text{ ns})$ 、即ち、 12740 ns である。7本のキャッシュラインを読み出し、1本を書き込むために要する時間は、より悪いケースであって、 1120 ns （全てのアクセスが同じDRAMバンクへのアクセスであるとして、 $8 \times 140 \text{ ns}$ ）である。その結果として、リソースが無制限であるならば、最大10画素を並列に処理することが可能である。ALUの個数が限定されている場合、最高で4画素を並列に処理することができる。従って、 7×7 カーネルを使用してコンボリユーシヨンを実行するために要する時間は、 0.18375 秒 （ $1500 \times 1000 \times 490 \text{ ns} / 4 = 183750000 \text{ ns}$ ）である。

【0655】

5×5 カーネル上で、各画素に要する時間は25サイクル、即ち、 250 ns である。各キャッシュラインは32画素を保持するので、メモリアクセスのために利用可能な時間は、 $((32 - 5 + 1) \times 250 \text{ ns})$ 、即ち、 7000 ns である。5本のキャッシュラインを読み出し、1本を書き込むために要する時間は、より悪いケースであって、 840

10

20

30

40

50

ns（全てのアクセスが同じDRAMバンクへのアクセスであるとして、 $6 \times 140 \text{ ns}$ ）である。その結果として、リソースが無制限であるならば、最大7画素を並列に処理することが可能である。ALUの個数が限定されている場合、最高で4画素を並列に処理することができる。従って、 5×5 カーネルを使用してコンポリューションを実行するために要する時間は、 0.09375 秒 （ $1500 \times 1000 \times 250 \text{ ns} / 4 = 93750000 \text{ ns}$ ）である。

【0656】

3×3 カーネル上で、各画素に要する時間は9サイクル、即ち、 90 ns である。各キャッシュラインは32画素を保持するので、メモリアクセスのために利用可能な時間は、 $(32 - 3 + 1) \times 90 \text{ ns}$ 、即ち、 2700 ns である。3本のキャッシュラインを読み出し、1本を書き込むために要する時間は、より悪いケースであって、 560 ns （全てのアクセスが同じDRAMバンクへのアクセスであるとして、 $4 \times 140 \text{ ns}$ ）である。その結果として、リソースが無制限であるならば、最大10画素を並列に処理することが可能である。ALU及び読み出し／書き込みイタレータの個数が限定されている場合、最高で4画素を並列に処理することができる。従って、 3×3 カーネルを使用してコンポリューションを実行するために要する時間は、 0.03375 秒 （ $1500 \times 1000 \times 90 \text{ ns} / 4 = 33750000 \text{ ns}$ ）である。この結果として、各出力画素は、計算をするためにカーネルサイズ／3サイクルを要する。実際のタイミングは以下の表に要約されている。

【0657】

【表59】

カーネルサイズ	出力画素計算 所要時間	1500×1000での 1チャンネル処理 所要時間	1500×1000での 3チャンネル処理 所要時間
3×3 (9)	3 サイクル	0.045 秒	0.135 秒
5×5 (25)	8 1/3 サイクル	0.125 秒	0.375 秒
7×7 (49)	16 1/3 サイクル	0.245 秒	0.735 秒

画像コンポジット

コンポジット処理は、最終画像における前景及び背景の適切な比率を制御するため、マット又はチャンネルを使用して前景画像を後景画像に加算することである。好ましくは、通常コンポジット処理及び付随コンポジット処理の二つのスタイルのコンポジット処理がサポートされる。二つのスタイルの規則は、

通常コンポジット処理：新しい値＝前景＋（背景－前景） α

付随コンポジット処理：新しい値＝前景＋（1－ α ）背景

である。

【0658】

相違点は、付随コンポジット処理の場合、前景はマットによって予め乗算されているのに対して、通常コンポジット処理の場合、そのようにされていないことである。コンポジットプロセスの一例は図83に示されている。

【0659】

アルファチャンネルは、レンジ0から1に対応した0から255の値をとる。

【0660】

通常コンポジット

通常コンポジットは、

背景＋（背景－前景） $\times \alpha / 255$

として実装される。

【0661】

$X / 255$ による除算は、 $257X / 65536$ によって近似される。コンポジットプロセスの実装例は図84に詳細に示され、以下の定数がソフトウェアによってセットされ

る。

【0662】

【表60】

定数	値
K_1	257

4個のイタレータが必要であるため、コンポジットプロセスは1画素当たり1サイクルを要し、ALUの半分しか利用しない。コンポジットプロセスは一つのチャンネルだけで実行される。3チャンネル画像を互いにコンポジットするためには、コンポジッターは、各チャンネルに1回ずつの合計3回実行されなければならない。

10

【0663】

フルサイズの単一チャンネルをコンポジットするための所要時間は、 0.015 秒（ $1500 \times 1000 \times 1 \times 10 \text{ ns}$ ）であり、3チャンネル全てをコンポジットするために 0.045 秒を要する。

【0664】

255 による除算を近似するため、 257 倍した後に 65536 で除算する。これは、1回の加算（ $256 \times x + x$ ）と、その結果の最後の16ビットの無視（丸めの目的を除く）とによっても実現可能である。

【0665】

図42に示されるように、コンポジットプロセスは、3個の順次読み出しイタレータ $351-353$ と、1個の順次書き出しイタレータ 355 とを必要とし、乗算器ALUと共にアダーALUと共にアダーを使用するマイクロコードとして実装される。コンポジット時間は1画素当たり1サイクル（ 10 ns ）である。付随コンポジット処理と通常コンポジット処理は別々のマイクロコードを必要とするが、1画素当たりの平均時間は同じである。

20

【0666】

コンポジット処理は単一チャンネルだけで実行される。一つの3チャンネル画像を互いにコンポジットするために、コンポジッターは、チャンネル毎に1回ずつ、併せて3回実行される。aチャンネルは各コンポジットに対して同一であるため、それは毎回読み出される必要がある。しかし、最良ケースでの 4×32 バイトのキャッシュラインの転送（読み出し又は書き込み）は 320 ns を要する。パイプラインは、平均で1画素コンポジット当たり1サイクルを与え、32画素をコンポジットするために32サイクル、即ち、 320 ns （ 100 MHz ）を要するので、チャンネルは実質的に無制限で読み出される。したがって、全チャンネルは、

30

$1500 / 32 \times 1000 \times 320 \text{ ns} = 15040000 \text{ ns} = 0.015$ 秒
でコンポジットされる。

【0667】

フルサイズの3チャンネル画像をコンポジットするための所要時間は 0.045 秒である。

40

【0668】

画像ピラミッド構築

ワーピング、タイリング、及びブラッシングのような幾つかの関数は、所定の画素のエリアの平均値を必要とする。所与の各エリアの値を計算するのではなく、これらの関数は、好ましくは、画像ピラミッドを利用する。既に図33に示されたように、画像ピラミッド360は、実質的に多重解像度画素マップである。原画像は1:1表現である。各次元での2:1によるサブサンプリングは、元のサイズの $1/4$ の画像を生成する。このプロセスは画像全体が単一画素によって表現されるまで続く。

【0669】

画像ピラミッドは原画像から構築され、原画像によって占められたサイズの $1/3$ （ $1/$

50

4 + 1 / 1 6 + 1 / 6 4 + . . .) を使う。原画像が 1 5 0 0 × 1 0 0 0 である場合、対応した画像ピラミッドは約 1 / 2 M B である。

【0 6 7 0】

画像ピラミッドは、コンボルのカーネルを各次元で 2 画素ずつ前進させるワン・イン・フォー入力画像に基づいて実行される 3 × 3 コンボルによって構築することができる。3 × 3 コンボルは、単純な 4 画素の平均化よりも高い精度を生じ、異なるピラミッドレベルの座標の差が 1 レベル毎に 1 ビットだけシフトすることによって得られるという付加的な利点がある。

【0 6 7 1】

ピラミッド全体の構築は、ピラミッドの各レベルで 1 回ずつピラミッドレベル構築関数を呼び出すソフトウェアループに依拠している。

10

【0 6 7 2】

ピラミッドの 1 レベルを生成するためのタイミングは、入力画像の解像度の (9 / 4) * (1 / 4) である。なぜならば、原画像の 1 / 4 のサイズの画像を生成しているからである。このようにして、1 5 0 0 × 1 0 0 0 画像に対して、

ピラミッドのレベル 1 を生成するためのタイミング = 9 / 4 * 7 5 0 * 5 0 0 = 8 4 3 7 5 0 サイクルであり、

ピラミッドのレベル 2 を生成するためのタイミング = 9 / 4 * 3 7 5 * 2 5 0 = 2 1 0 9 3 8 サイクルであり、

ピラミッドのレベル 3 を生成するためのタイミング = 9 / 4 * 1 8 8 * 1 2 5 = 5 2 7 3 5 サイクルであり、

20

以下同様に続く。

【0 6 7 3】

総時間は、原画像 1 画素あたりに 3 / 4 サイクル (画像ピラミッドは原画像サイズの 1 / 3 であり、各画素は計算に 9 / 4 サイクルを要し、即ち、(1 / 3) * (9 / 4) = 3 / 4) である。1 5 0 0 × 1 0 0 0 画像の場合、(1 0 0 M H z) で 1 1 2 5 0 0 0 サイクル、即ち、0 . 0 1 1 秒である。このタイミングは単一色チャンネルに対するタイミングであり、3 色チャンネルは 0 . 0 3 4 秒の処理時間を要する。

【0 6 7 4】

汎用データ駆動型画像ワーパー

30

A C P 3 1 は、入力画像の画像ワーピング操作を実行可能である。画像ワーピングの原理は理論的に周知である。ワーピングのプロセスに関する一つの完全な教科書は、G e r o g e

W o l b e r g による ' & a p s ; D i g i t a l I m a g e W a r p i n g ' & a p s ; , I E E E C o m p u t e r S o c i e t y P r e s s , L o s A l a m i t o s ,

C a l i f o r n i a ,

1 9 9 0 である。ワーピングプロセスは、A r t c a r d 9 によって供給されたデータの一部を形成するワープマップを使用する。ワープマップは、必要条件に応じて任意的に寸法を決めることができ、入力画素の出力画素へのマッピングの情報を提供する。残念なことに、任意的に寸法を決められたワープマップの利用は、画像ワーパーによって解決されるべき多数の問題を生じる。

40

【0 6 7 5】

図 8 5 を参照すると、A × B 次元のワープマップ 3 6 5 は、ある大きさ (例えば、0 から 2 5 5 の 8 ビット値) の配列値により構成され、この配列値は理論的な入力画像の座標を表現し、理論的な入力画像の座標は同じ配列座標インデックスを有する対応した「理論的な」出力画像へマップされる。残念ながら、出力画像、例えば、出力画像 3 6 6 は固有の次元 C × D を有し、これは、固有の次元 E × F を有する入力画像とは全く異なる。したがって、出力画像 3 6 6 が、出力画素毎に、対応した入力画像のエリア又は領域を決定できるように、ワープマップ 3 6 5 のマッピングを容易に行えるようにする必要がある。出力

50

画素色データは、対応した入力画像のエリア又は領域から構築される。出力画像 3 6 6 内の各出力画素に対して、最初に、ワーブマップ 3 6 5 から対応したワーブマップ値を決定することが必要である。このため、出力画像画素がワーブマップテーブル 3 6 5 内の小数位置にマッピングされるときに、周辺ワーブマップ値をバイリニア補間することが必要である。このプロセスの結果は、ワーブマップ 3 6 5 内の各データ値のサイズによって寸法が決められる「理論的な」画像内での入力画像画素の位置を与える。これらの値は、理論的な画像を対応した実際の入力画像 3 6 7 へマッピングするため再度スケーリングしなければならない。

【0 6 7 6】

実際の値を決定するため、出力画像画素はエイリアシング効果を回避するように選ばれるべきであり、最終的な出力画像画素値に寄与する入力画像画素 3 6 7 の領域を決定するために隣接した出力画像画素を調べる必要がある。この点に関して、以下でより明らかにされるように、画像ピラミッドが利用される。

【0 6 7 7】

画像ワーパーは画像をワーブさせるため以下の幾つかのタスクを実行する：

- －出力画像サイズに一致するようにワーブマップをスケーリングする；
- －各出力画素に表現される入力画像画素の領域のスパンを決定する；
- －入力画像ピラミッドからトライリニア補間によって最終的な出力画素値を計算する。

【0 6 7 8】

ワーブマップスケーリング

前述の通り、データ駆動型ワーブでは、ワーブマップは、出力画素毎に、対応した入力画像マップの中心を記述することが必要である。前述のように単一ワーブマップを用いる代わりに、インターリーブされた x 及び y 値情報を格納すると、X 座標及び Y 座標を別個のチャンネルとして取り扱うことが可能である。

【0 6 7 9】

その結果として、好ましくは、二つのワーブマップが設けられ、X ワーブマップは X 座標のワーピングを表し、Y ワーブマップは Y 座標のワーピングを示す。前述のように、ワーブマップ 3 6 5 は、スケーリング中の画像とは異なる空間解像度でもよい（例えば、3 2 × 3 2 ワーブマップ 3 6 5 は、1 5 0 0 × 1 0 0 0 画像 3 6 6 のワーブを適切に記述する）。その上、ワーブマップは、ワーブ中の画像のサイズに対応した 8 又は 1 6 ビット値によって表現可能である。

【0 6 8 0】

所定のワーブマップから入力画像空間に点を生成するために幾つかのステップが含まれる：

1. 出力画像に対してワーブマップ内に対応した位置を決定する；
2. 次のステップのため、ワーブマップから値をフェッチする（これは、ワーブマップがほんの 8 ビット値である場合に、解像度ドメインにおけるスケーリングを必要とする）；
3. 実際の値を決定するためワーブマップのバイリニア補間を行う；
4. 入力画像ドメインに対応するように値をスケーリングする。

【0 6 8 1】

最初のステップは、出力画像における現在の X / Y 座標にスケール倍率（X と Y で異なる場合がある）を乗算することにより実行される。例えば、出力画像が 1 5 0 0 × 1 0 0 0 であるならば、ワーブマップは 1 5 0 × 1 0 0 であり、X と Y の両方は 1 / 1 0 によってスケーリングされている。

【0 6 8 2】

ワーブマップから値をフェッチするためには、二つのルックアップテーブルにアクセスすることが必要である。一方のルックアップテーブルは、ルックアップテーブルから 8 又は 1 6 ビットのエントリーを読み出し、常に 1 6 ビット値を返す（元の値が 8 ビットしかない場合、上位 8 ビットをクリアする）。

【0 6 8 3】

パイプラインの次のステップは、参照されたワーブマップ値をバイリニア補間することである。

【0684】

最後に、バイリニア補間からの結果がスケーリングされ、ワーブされるべき画像と同じドメインに収容される。このようにして、ワーブマップのレンジが0から255である場合、Xを1500/255でスケーリングし、Yを1000/255でスケーリングする。

【0685】

補間プロセスは図86に示されているようなプロセスであり、以下の定数がソフトウェアによってセットされる。

【0686】

【表61】

定数	値
K ₁	Xscale (0-ImageWidth から 0-WarpmapWidth へのスケール)
K ₂	Yscale (0-ImageHeight から 0-WarpmapHeight へのスケール)
K ₃	XrangeScale (ワーブマップレンジ (例えば、0 - 255) から 0-ImageWidth へのスケール)
K ₄	YrangeScale (ワーブマップレンジ (例えば、0 - 255) から 0-ImageHeight へのスケール)

以下のルックアップテーブルが使用される。

【0687】

【表62】

ルックアップ	サイズ	詳細
LU ₁ 及び LU ₂	WarpmapWidth × WarpmapHeight	ワーブマップルックアップ。 [X, Y] が与えられるとバイリニア補間のために必要な4エントリーが返される。エントリーが8ビットしかない場合でも、16ビット (上位8ビットは0) として返される。 転送時間は1エントリー当たり2バイトで4エントリーである。 総時間は、二つのルックアップが使用されるとき8サイクルである。

スパン計算

ワーブマップ365からの点は入力画像367内の画素領域の中心の位置を示す。隣接した出力画像画素の入力画像画素の間の距離は領域のサイズを示し、この距離は、スパン計算によって近似することができる。

【0688】

図87を参照すると、ワーブマップP1の所定の現在点に対して、同じライン上の前のポイントはP0であり、前のラインの同じ位置にあるポイントはP2である。P1とP0の間、並びに、P1とP2の間で、X及びYの絶対距離を決定する。X又はYの最大距離は、実際の形状を近似する正方形であるスパンになる。

【0689】

好ましくは、点は、垂直方向ストリップの出力順に処理され、P0はストリップ内の同じ線で先行する点であり、P1がストリップ内の最初の点であるとき、P0は、前のストリップの対応した線の最後の点を表す。P2は同一ストリップ内の前の線の点であるため、32エントリーの履歴バッファに保持され得る。スパン計算プロセスの原理は図88に示され、そのプロセスの詳細は図89に示されている。

【0690】

以下のDRAMのFIFOが使用される。

【 0 6 9 1 】

【表 6 3】

ルックアップ	サイズ	詳 細
FIFO ₁	8ImageWidth バイト [ImageWidth×1 エントリー当たり 32 ビ ットの 2 エントリー]	P2 履歴／ルックアップ (X と Y の両方が同じ FIFO にある) P ₁ は FIFO に入れられ、P2 として後続の行で同じ画素に 再度取り出される。 転送時間は 4 サイクル (2×32 ビット、16 ビットあたりに 1 サイクル)

10

3 2 ビット精度のスパン履歴が保持されるので、1 5 0 0 画素幅の画像がワープされる場
合、1 2 0 0 0 バイトの一時記憶が必要になる。

【 0 6 9 2 】

スパン 3 6 4 の計算は、二つのアダー A L U (一方はスパン計算用であり、もう一方は P
0 及び P 2 の履歴をループしカウントするためのものである) を使用し、以下のように 7
サイクルを要する。

【 0 6 9 3 】

【表 6 4】

サイクル	動 作
1	$A = \text{ABS}(P1_x - P2_x)$ P1 _x を P2 _x 履歴に保存
2	$B = \text{ABS}(P1_x - P0_x)$ P1 _x を P0 _x 履歴に保存
3	$A = \text{MAX}(A, B)$
4	$B = \text{ABS}(P1_y - P2_y)$ P1 _y を P2 _y 履歴に保存
5	$A = \text{MAX}(A, B)$
6	$B = \text{ABS}(P1_y - P0_y)$ P1 _y を P0 _y 履歴に保存
7	$A = \text{MAX}(A, B)$

20

30

履歴バッファ 3 6 5、3 6 6 は、キャッシュ型の D R A M である。「前のライン」(P 2
履歴の場合) バッファ 3 6 6 は、3 2 エントリーのスパン精度である。「前の点」(P 0
履歴の場合) バッファ 3 6 5 は、(ストリップの線の点 1 から 3 1 の計算のため) 殆どの
場合に使用される 1 個のレジスタを必要とし、D R A M にバッファされた履歴値の組はス
トリップの線の点 0 の計算に使用される。

40

【 0 6 9 4 】

スパン履歴の 3 2 ビット精度は、P 2 履歴を保持するため 4 本のキャッシュラインと、P
0 履歴のための 2 本のキャッシュラインと、を必要とする。P 0 の履歴は、3 2 画素から
なる 8 本の線毎に 1 回ずつ、(ImageHeight×4) バイトの一時記憶空間に書き込まれ、
読み出される。このようにして、1 5 0 0 画素の高さの画像がワープされるとき、6 0 0
0 バイトの一時記憶が必要であり、全部で 6 本のキャッシュラインが必要である。

【 0 6 9 5 】

トライリニア補間

50

平均化されるべき入力画像からエリアの中心及びスパンを決定した後、ワーブプロセスの最終部分は、出力画素の値を決定することである。単一の出力画素は、理論的に入力画像全体によって表現できるので、出力画素に寄与する入力画像の特定のエリアを実際に読み出し平均化するためには、非常に時間を消費する可能性がある。その代わりとして、入力画像の画像ピラミッドを使用することにより画素値を近似することができる。

【0696】

スパンが1以下である場合、所与の座標の周辺で原画像の画素を読み出し、バイリニア補間を実行するだけでよい。スパンが1よりも大きい場合、二つの適切なレベルの画像ピラミッドを読み出し、トライリニア補間を実行する必要がある。二つのレベルの画像ピラミッドの間にリニア補間を実行することは、厳密に正確ではないが、許容可能な結果を与える（それは結果画像にブラー効果が生じる側で判断を誤る）。

10

【0697】

図90を参照すると、一般的には、あるスパン「s」が与えられた場合に、 $\ln_2 s$ (370) 及び $\ln_2 s + 1$ (371) によって与えられる画像ピラミッドレベルを読み出すことが必要である。 $\ln_2 s$ は、sの上位ビットセットを復号化するだけである。ピラミッドの二つのレベル370、371の各々で画素値の値を決定するためバイリニア補間を実行し、レベル間で補間を実行しなければならない。

【0698】

図91に示されるように、最終的な出力値373を得るため、ピラミッドレベル間で補間を行う前に、最初に各画素ピラミッドレベルのX及びYに関して補間を行うことが必要である。

20

【0699】

画像ピラミッドアドレスモードは、ピラミッドレベルs及びs+1上の(x, y)における画素座標に対するアドレスを生成する。各レベルの画像ピラミッドは、xの方向に連続した画素を格納する。したがって、xに関する読み出しはキャッシュヒットする可能性が高い。

【0700】

適度なキャッシュヒットのコヒーレンスは、出力画像の局所領域が典型的に入力画像と局所的にコヒーレントである（おそらく、スケールは異なるが、同じスケールでコヒーレントである）ときに獲得される。入力画像と出力画像との関係を知ることはできないが、出力画素は、キャッシュコヒーレンスを最大限に利用するため、（垂直ストリップイタレータによって）確実に垂直ストリップに書き込まれる。

30

【0701】

トライリニア補間は、パイプラインとして4個の乗算ALU及び4個全てのアダーALUを使用し、メモリアクセスの必要がない場合を想定すると、平均的に2サイクル程度で完了し得る。しかし、全ての補間値は画像ピラミッドから得られるので、補間速度はキャッシュコヒーレンスに完全に依存する（その他のユニットはワーブマップスケーリング及びスパン計算を実行するためビジーであることは言うまでもない）。できるだけ多数のキャッシュラインを画像ピラミッド読み出しのために利用できるべきである。最良の速度は、2個の乗算ALUを使用して8サイクルである。

40

【0702】

出力画素は、二つのキャッシュラインを使用する垂直ストリップ書き込みイタレータによってDRAMに書き込まれる。この速度は、従って、1出力画素当たり8サイクルの最小値で制限される。ワーブマップのスケーリングに必要なサイクルが8サイクル以下であるならば、全体的な速度は変化しない。さもなければ、スループットはワーブマップをスケーリングするために要する時間である。殆どの場合に、ワーブマップは写真のサイズと適合するまで拡大される。

【0703】

1画素当たりに必要なサイクル数が8サイクル以下のワーブマップをスケーリングする場合、画像の1個のカラーコンポーネントを変換するための所要時間は0.12秒(150

50

0 * 1 0 0 0 * 8 サイクル * 1 サイクル 当たり に 1 0 n s) である。

【0704】

ヒストグラムコレクタ

ヒストグラムコレクタは、入力として画像チャンネルを取得し、出力としてヒストグラムを生成するマイクロコードプログラムである。チャンネルの画素の各々は、レンジ 0 - 255 に収まる値をとる。その結果として、ヒストグラムテーブルには全部で 256 画素が存在し、各エントリは 32 ビットであり、1500 × 1000 の全画像のカウンタを十分に収容することができる。

【0705】

図 9 2 に示されるように、ヒストグラムは全画像の要約を表現するので、順次読み出しイタレータ 378 は入力として十分にである。ヒストグラム自体は、完全にキャッシュされ、32 個のキャッシュライン (1 K) を必要とする。

【0706】

マイクロコードは二つのパスを有し、初期化パスでは、全てカウンタを 0 にセットし、次に、「カウンタ」ステージは、画像から読み出された各画素の適切なカウンタをインクリメントする。第 1 のステージは、アドレスユニットと単一のアダー A L U とを必要とし、ヒストグラムテーブル 377 のアドレスを初期化する。

【0707】

【表 6 5】

相対 マイクロコードアドレス	アドレスユニット A = ヒストグラムのベースアドレス	アダーユニット 1
0	0 を $A + (\text{Adder1.Out}_1 \ll 2)$ に書き込み	$\text{Out}_1 = A$ $A = A - 1$ BNZ 0
1	処理のリセット	処理のリセット

第 2 のステージは、画像からの実際の画素を処理し、4 個のアダー A L U を使用する。

【0708】

【表 6 6】

	アダー-1	アダー-2	アダー-3	アダー-4	アドレスユニット
1	$A = 0$			$A = -1$	
2 BZ 2	$\text{Out}_1 = A$ $A = \text{pixel}$	$Z = \text{Adder1.Out}_1$ $Z = \text{pixel} - \text{Adder1.Out}_1$	$A = \text{Adr.Out}_1$	$A = A + 1$	$\text{Out}_1 = (A + (\text{Adder1.Out}_1 \ll 2))$ から 4 バイト読み出し
3		$\text{Out}_1 = A$	$\text{Out}_1 = A$	$\text{Out}_1 = A$ $A = \text{Adder3.Out}_1$	Adder4.Out_1 を $(A + (\text{Adder2.Out}_1 \ll 2))$ へ書き込み
4					Adder4.Out_1 を $(A + (\text{Adder2.Out}_1 \ll 2))$ へ書き込み キャッシュを消去

アダー 2 のサイクル 2 からの零フラグは、入力画素が同じである間、マイクロコードアドレス 2 に留まるため使用される。入力画素が変化するとき、新しいカウンタはマイクロコードアドレス 3 に書き出され、処理はマイクロコードアドレス 2 で再開する。マイクロコードアドレス 4 は最後に使用され、それ以上読み出される画素は存在しない。

【0709】

ステージ 1 は、256 サイクル、即ち、2560 n s を要する。ステージ 2 は、画素の値に応じて変わる。ルックアップテーブル置換のための最悪ケースの時間は、あらゆる画素

がその近傍と同じではない場合に、1画像画素当たり2サイクルである。単一カラーのルックアップに要する時間は0.03秒（ $1500 \times 1000 \times 1$ 画素当たり2サイクル $\times 1$ サイクル当たり10ns = 3000000ns）である。3個のカラーコンポーネントに要する時間はこの量の3倍、即ち、0.09秒である。

【0710】

色変換

色変換は、

ルックアップテーブル置換、及び

色空間変換

の二つの主要な方法で実現される。

10

【0711】

ルックアップテーブル置換

図86に示されるように、画素の色を変換するための最も簡単な方法の一つは、任意に複雑な変換関数をルックアップテーブル380にエンコードすることである。画素のコンポーネントカラー値は、画素の新しいコンポーネント値をルックアップ381するため使用される。順次読み出しイタレータから読み出された画素毎に、その新しい値が新カラーテーブル380から読み出され、順次書き込みイタレータ383へ書き込まれる。入力画像は、メモリ帯域幅を効率的に利用するため、半分ずつ同時に処理される。以下のルックアップテーブルが使用される。

【0712】

【表67】

20

ルックアップ	サイズ	詳細
LU ₁	256 エントリー 1 エントリー当たり8ビット	[X] 置換 テーブルはXの最上位8ビットでインデックス付け 得られた8ビットは固定小数点0:8として取扱

総プロセスは、二つの順次読み出しイタレータ及び二つの順次書き込みイタレータを必要とする。二つの新カラーテーブルは、256バイト（1バイトからなる256個のエントリー）を保持するため、各々に8個のキャッシュラインが必要である。

30

【0713】

したがって、ルックアップテーブル置換に要する平均時間は1画像画素当たり1/2サイクルである。単一カラーのルックアップに要する時間は0.0075秒（ $1500 \times 1000 \times 1$ 画素当たり1/2サイクル $\times 1$ サイクル当たり10ns = 750000ns）である。三つのカラーコンポーネントのための所要時間はこの量の3倍、即ち、0.0225秒である。各カラーコンポーネントは、ソフトウェアの制御下で次々に処理される。

【0714】

色空間変換

色空間変換は、色空間の間を動くときに限り必要である。CCD画像はRGB色空間で撮影され、印刷はCMY色空間で行われ、一方、ACP31のクライアントは、Lab色空間で画像を処理する可能性が高い。入力色空間チャンネルの全ては、典型的に、各出力チャンネルのコンポーネント値を決定するために要求される。このような論理的プロセスが図94の385に示されている。

40

【0715】

簡潔に言うと、Lab、RGB及びCMYの間の変換は、かなり簡単である。しかし、特定の装置の個別の色プロファイルは著しく変化し得る。その結果として、将来のCCD、インク、及びプリンタを可能にさせるため、ACP31は、色空間変換ルックアップテーブルからのトライリニア補間によって色空間変換を実行する。

【0716】

色コヒーレンスは、線ベースではなく、エリアベースである傾向がある。トライリニア補

50

間ルックアップ中にキャッシュコヒーレンスを補助するため、画像を垂直ストリップで処理することが最良である。このようにして、読み出しイタレータ 386-388 及び書き出しイタレータ 389 は、垂直ストリップイタレータである。

【0717】

トライリニア色空間変換

各出力カラーコンポーネントに対して、入力色空間を出力色空間へマッピングする単一の 3 次元テーブルが必要になる。例えば、C C D 画像を R G B から L a b へ変換するため、C C D の物理的特性に較正された 3 個のテーブル、即ち、

R G B → L

R G B → a

R G B → b

が必要になる。

【0718】

L a b から C M Y へ変換するため、インク／プリンタの物理的特性に較正された 3 個のテーブル、即ち、

L a b → C

L a b → M

L a b → Y

が必要になる。

【0719】

8 ビット入力カラーコンポーネントは、変換テーブルへのインデックス付けのため、固定小数点数 (3 : 5) として取り扱われる。整数の 3 ビットはインデックスを与え、小数の 5 ビットは補間として使用される。3 ビットは 8 個の値を与えるので、3 次元で 512 エントリー (8 × 8 × 8) が得られる。各エントリーのサイズは 1 ビットであり、1 テーブル当たり 512 バイトが必要である。

【0720】

したがって、色空間変換プロセスは図 95 に示されるように実装され、以下のルックアップテーブルが使用される。

【0721】

【表 68】

ルックアップ	サイズ	詳細
LU ₁	8 × 8 × 8 エントリー 512 エントリー 1 エントリー当たり 8 ビット	[X, Y, Z] 変換 テーブルは X, Y 及び Z の最上位 3 ビットでインデックス付け トライリニアインデックスアドレスユニットから 8 ビットが返却 得られた 8 ビットは固定小数点 0 : 8 として取扱 転送時間は 1 エントリー当たり 1 バイトで 8 エントリー

トライリニア補間は、8 個の値の間の補間を返す。ルックアップから各 8 ビット値を返すために 1 サイクルを要し、全部で 8 サイクルを要する。トライリニア補間は、1 サイクル当たり 2 個の乗算 A L U が使用されたとき、8 サイクルを使用する。一般的なトライリニア補間情報はこの文書の A L U セクションに記載されている。ルックアップテーブル用の 512 バイトは 16 個のキャッシュラインに詰め込まれる。

【0722】

画像の単一のカラーコンポーネントを変換するための時間は 0.105 秒 (1500 × 1000 × 7 サイクル × 1 サイクル当たり 10 ns) である。3 コンポーネントを変換するため、0.415 秒を要する。都合良く、プリントアウト用の色空間変換は、プリントアウト中にオンザフライ方式で行われる。

【0723】

カラーコンポーネントが別々に変換される場合、それらは入力色空間コンポーネントを上書きしなければならない。なぜならば、入力色空間からの全てのカラーコンポーネントは各コンポーネントを変換するために必要になるからである。

【0724】

補間を実行するために1個の乗算ユニットだけが使用されるので、Lab->CMY変換の全体を単一パスとして行うことが可能である。このためには、三つの垂直ストリップ読み出しイタレータと、三つの垂直ストリップ書き込みイタレータと、が必要であり、3個の変換テーブルに同時にアクセスする。その場合、入力画像へ書き戻し、余分なメモリを使用しないことも可能である。しかし、3個の変換テーブルへのアクセスは、各々に対するキャッシュ処理の1/3に相当し、プロセス全体では高いレイテンシーを生じることになる。

10

【0725】

アフィン変換

写真を用いて画像をコンポジットする前に、画像を回転、スケーリング、及び並進することが必要である。画像が並進だけされている場合、直接的なサブピクセル平行移動関数を使用する方が高速になる。しかし、回転、拡大、及び並進は、単一のアフィン変換に組み込むことができる。

【0726】

汎用アフィン変換は、加速関数として組み込みことが可能である。アフィン変換は2次元に制限され、縮小される場合、入力画像はスケール関数によって予めスケーリングされるべきである。汎用アフィン変換を備えることにより、出力画像を同時に1ブロックずつ構築することが可能になり、画像に対して多数の変換を実行するために要する時間を削減できる。なぜならば、多数の変換の全てを同時に適用できるからである。

20

【0727】

変換行列はクライアントによって与えられる必要があり、行列は、望まれる変換の逆行列でなければならない。即ち、その行列を出力画素座標に適用することによって入力座標が得られる。

【0728】

2次元行列は、通常、以下の3×3配列、即ち、

【0729】

30

【数1】

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

40

によって表現される。

【0730】

第3列は常に[0, 0, 1]であるため、クライアントはそれを指定しなくてもよい。その代わりに、クライアントはa, b, c, d, e, fを指定する。

【0731】

左上画像画素の座標が(0, 0)として表される入力画像内の座標(x, y)が与えられた場合、入力座標は、(ax + cy + e, bx + dy + f)によって指定される。入力座標が決定されると、入力画像は、その画素値に達するためサンプリングされる。入力画像画素のバイリニア補間は、計算された座標における画素の値を決定するため使用される。アフィン変換は平行線を保存するので、画像は、最高の平均入力画像キャッシュコヒーレ

50

ンスを得るため 32 画素幅の出力垂直ストリップで処理される。

【0732】

3 個の乗算 A L U は、バイリニア補間を 2 サイクル内で実行するために必要である。乗算 A L U 1 及び 2 は、線 Y 及び線 Y + 1 のそれぞれについて X に関してリニア補間を実行し、乗算 A L U 3 は、乗算 A L U 1 及び 2 によって出力された値の間で Y に関するリニア補間を実行する。

【0733】

X に関して出力線を横切って右へ移動するとき、2 個のアダー A L U は、「a」を現在の X 値に加算し、「b」を現在の Y 値に加算することによって、実際の入力画像座標を計算する。次の線（最大で 32 画素を処理した後の垂直ストリップ内の次の線、又は新しい垂直ストリップの最初の線）へ進んだとき、X 及び Y を所与のブロックに対する予め計算されたスタート座標値定数に更新する。

【0734】

入力座標を計算するプロセスは図 96 に示され、同図では、以下の定数がソフトウェアによってセットされる。

【0735】

画素計算

入力画像座標が得られると、入力画像をサンプリングする必要がある。ルックアップテーブルは、バイリニア補間の準備として指定された座標における値を返すため使用される。基本プロセスは図 97 に示され、以下のルックアップテーブルが使用される。

【0736】

【表 69】

ルックアップ	サイズ	詳 細
LU ₁	画像幅×画像高 1 エントリー当たり 8 ビット	バイリニア画像ルックアップ [X, Y] テーブルは X 及び Y の整数部によってインデックス付け 1 サイクル当たり 2 個ずつ、バイリニアインデックスアドレスユニットから 4 エントリーを返却 各 8 ビットエントリーを固定小数点 8:0 として取扱 転送時間は 2 サイクル(FIFO の 216 ビットエントリーは 48 ビットエントリーを保持)

アフィン変換は、全部で 4 個の乗算 A L U 及び全部で 4 個のアダー A L U を必要とし、良好なキャッシュコヒーレンスがある場合、1 出力画素当たり平均 2 サイクルでアフィン変換を実行できる。このタイミングは優れたキャッシュコヒーレンスを前提とし、この前提は画像に歪みが無い場合には成立する。最悪ケースでは、画像がかなり歪み、重要な V a r k スクリプトが含まれている可能性は低い。

【0737】

128×128 画像を変換するために要する時間は、0.00033 秒（32768 サイクル）である。これが 4 チャンネル（a チャンネルを含む）を備えたクリップ画像である場合、必要な総時間は 0.00131 秒（131072 サイクル）である。

【0738】

垂直ストリップ書き込みイタレータは画素を出力するため必要である。読み出しイタレータは不要である。しかし、アフィン変換アクセラレータは、入力画像画素にアクセスするために要する時間によって制約されるので、できるだけ多数のキャッシュラインが入力画像からの画素の読み出しに割り付けられるべきである。少なくとも 32 が利用可能であり、好ましくは、64 以上が利用可能である。

【0739】

スケーリング

スケーリングは、基本的に画像の再サンプリングである。画像の拡大はアフィン変換関数を用いて実行できる。縮小を含む一般化された画像のスケーリングは、ハードウェア加速

型スケール関数によって行われる。スケーリングはX及びYに関して独立に行われるので、異なるスケール倍率を各次元で使うことができる。一般化されたスケールユニットは、位置合わせに関して、アフィン変換スケール関数と適合しなければならない。

【0740】

一般化されたスケーリングプロセスは図98に示されている。Xに関するスケールは図99に示されるようにFantの再サンプリングアルゴリズムによって達成される。

【0741】

以下の定数がソフトウェアによってセットされる。

【0742】

【表70】

10

定 数	値
K_1	Xの出力画素に寄与する入力画素の個数
K_2	$1/K_1$

以下のレジスタは一時変数を保持するため使用される。

【0743】

【表71】

20

変 数	値
Latch ₁	未使用のままの入力画素の量 (1で開始しデクリメント)
Latch ₂	現在出力画素に寄与するため残っている入力画素の量 (K_1 で開始しデクリメント)
Latch ₃	次の画素 (Xの中で)
Latch ₄	現在画素
Latch ₅	出力画素のアクキュムレータ (スケーリング無し)
Latch ₆	Xでスケーリングされた画素 (出力)

Yに関するスケールプロセスは図100に示され、X画素の順番の処理を考慮するためFantの再サンプリングアルゴリズムの僅かに変更されたバージョンによっても実現される。

30

【0744】

以下の定数がソフトウェアによってセットされる。

【0745】

【表72】

定 数	値
K_1	Yの出力画素に寄与する入力画素の個数
K_2	$1/K_1$

40

以下のレジスタが一時変数を保持するため使用される。

【0746】

【表 7 3】

変 数	値
Latch ₁	未使用のままの入力画素の量 (1 で開始しデクリメント)
Latch ₂	現在出力画素に寄与するため残っている入力画素の量 (K ₁ で開始しデクリメント)
Latch ₃	次の画素 (Y の中で)
Latch ₄	現在画素
Latch ₅	Y でスケーリングされた画素 (出力)

10

以下の D R A M F I F O が使用される。

【 0 7 4 7 】

【表 7 4】

ルックアップ	サイズ	詳 細
FIFO ₁	ImageWidth _{OUT} エントリー 1 エントリー当たり 8 ビット	X でスケーリング済みの 1 列の画像画素 1 サイクル転送時間
FIFO ₂	ImageWidth _{OUT} エントリー 1 エントリー当たり 16 ビット	X でスケーリング済みの 1 列の画像画素 2 サイクル転送時間 (1 サイクル当たり 1 バイト)

20

画像モザイク処理

画像のモザイク処理はタイリングの一形態である。モザイク処理は、特別に設計された「タイル」を、水平方向及び垂直方向に複数回、第 2 の（通常はより大きい）画像空間へコピーする処理を含む。モザイク処理されたとき、小さいタイルは継ぎ目のないピクチャーを形成する。この一例は、レンガ壁の区画の小さいタイルである。それは、モザイク処理されたときに、完全なレンガ壁を形成するように設計される。尚、モザイク処理には、スケーリング、又はサブピクセル平行移動が含まれないことに注意すべきである。

30

【 0 7 4 8 】

モザイク処理を行うために最もキャッシュコヒーレントな方法は、画像をライン毎に順次に出力し、同じ入力画像のラインを、そのラインの区間の間、繰り返すことである。ラインが終了したとき、入力画像は次のラインへ進められるべきである（出力ラインをはさんで複数回それを繰り返すべきである）。

【 0 7 4 9 】

モザイク関数の概要は図 1 0 1 の 3 9 0 に示されている。順次読み出しイタレータ 3 9 2 は入力タイルの単一ラインを連続して読み出すようにセットアップされる（スタートラインは 0 であり、エンドラインは 1 である）。各入力画素は、書き込みイタレータ 3 9 3 - 3 9 5 の三つ全てに書き込まれる。アダー A L U のカウンタ 3 9 7 は、出力ライン中の画素数をカウントダウンし、ラインの最後にシーケンスを終了する。

40

【 0 7 5 0 】

ラインの処理の最後に、小さいソフトウェアルーチンは、マイクロコード及び順次読み出しイタレータ（これは、F I F O をクリアし、タイルのライン 2 を繰り返す）を再始動する前に、順次読み出しイタレータのスタートライン及びエンドラインレジスタを更新する。書き込みイタレータ 3 9 3 - 3 9 5 は更新されず、出力画像の対応した部分へ書き込み続けるだけである。正味の効果として、タイルが出力ラインをはさんで 1 ライン繰り返され、タイルが垂直方向に繰り返される。

【 0 7 5 1 】

このプロセスはメモリ帯域幅を完全には使用しない。なぜならば、入力画像におけるキャ

50

ッシュコヒーレンスは優れているが、モザイク処理を他のサイズのタイルと共に機能させることができないからである。このプロセスは、1個のアダーALUを使用する。3個の書き込みイタレータ393-395の各々が、画像の1/3に書き込む（画像をタイルサイズの境界に分割する）ならば、モザイクプロセスの全体は1出力画像画素当たり1/3サイクルの平均速度で行われる。1500×1000の画像の場合、これは、0.005秒（5000000ns）と一致する。

【0752】

サブピクセルトランスレータ

画像を背景とコンポジットする前に、X及びYの両方に関してサブピクセル量で画像を平行移動させることが必要である。サブピクセル平行移動は、各次元に1画素ずつ画像のサイズを増加することができる。画像の外側の領域の値は、定数値（例えば、黒色）又はエッジ画素繰り返しのようなクライアントによって決定される値である。典型的に、黒色を使用する方がよい。

10

【0753】

サブピクセル平行移動プロセスは図102に示されている。所定の次元のサブピクセル平行移動は、

$$\text{Pixel}_{\text{out}} = \text{Pixel}_{\text{in}} * \\ (1 - \text{Translation}) + \text{Pixel}_{\text{in}-1} * \\ \text{Translation}$$

によって定義される。

20

【0754】

これは、補間の形式、即ち、

$$\text{Pixel}_{\text{out}} = \text{Pixel}_{\text{in}-1} + \\ (\text{Pixel}_{\text{in}} - \text{Pixel}_{\text{in}-1}) \\ * \text{Translation}$$

として表現することができる。

【0755】

単一の乗算ALU及び単一のアダーALUを組み合わせて使用する単一（又は平均）サイクル補間エンジンの実装は簡単である。X及びYの両方に関するサブピクセル平行移動は、二つの補間エンジンを必要とする。

30

【0756】

Yに関するサブピクセル平行移動を行うため、二つの順次読み出しイタレータ400、401が必要になり（一方は、同じ画像から他の画像よりも進んでいるラインを読み出す）、一つの順次書き込みイタレータ403が必要になる。

【0757】

第1の補間エンジン（Yに関する補間）は、二つのストリームからのデータペアを受け取り、それらの間を線形に補間する。第2の補間エンジン（Xに関する補間）は、単一の1次元ストリームとしてそのデータを受け取り、値の間を線形に補間する。両方のエンジンは平均として1サイクルで補間を行う。

【0758】

各補間エンジン405、406は、平均的に1出力画素当たり1サイクルでサブピクセル平行移動を実行する能力を備えている。したがって、全体的な時間は1出力画素当たり1サイクルであり、2個の乗算ALUと2個のアダーALUが必要である。

40

【0759】

サブピクセル平行移動関数から32画素を出力するために要する時間は、平均で320ns（32サイクル）である。これは、DRAMへの完全な4個のキャッシュラインアクセスに十分な時間であり、3個の順次イタレータの使用はタイミング原価以内に十分に収まる。

【0760】

画像をサブピクセル平行移動するために要する総時間は、したがって、出力画像の1画素

50

当たり 1 サイクルである。サブピクセル平行移動されるべき典型的な画像は、サイズ 1 2 8 * 1 2 8 のタイルである。出力画像サイズは 1 2 9 * 1 2 9 である。このプロセスは、 $1\ 2\ 9 * 1\ 2\ 9 * 1\ 0\ n\ s = 1\ 6\ 6\ 4\ 1\ 0\ n\ s$ である。

【0761】

画像タイル貼り関数もサブピクセル平行移動アルゴリズムを使用するが、サブピクセル平行移動処理されたデータを書き出す必要はなく、そのデータを更に処理する。

【0762】

画像タイルタイラー

画像をタイル貼りする上位レベルアルゴリズムはソフトウェアで実行される。タイルの配置が決定されると、適切な着色タイルを合成しなければならない。各タイルの画像への実際のコンポジット処理は、マイクロコード化された A L U によってハードウェアで行われる。タイルのコンポジット処理は、背景画像へのテクスチャ貼り付けとカラー貼り付けの両方を含む。ある種のケースでは、背景に加えられる実際のテクスチャ量と、予定されたテクスチャ量とを比較し、貼り付けられるカラーをスケーリングするためこれを使用することが望ましい。これらのケースでは、テクスチャを先に貼り付けることが必要である。

【0763】

カラー貼り付け機能及びテクスチャ貼り付け機能はある程度独立しているので、これらはサブ関数に分離される。

【0764】

種々のテクスチャスタイル及びカラーリングスタイルのための 4 チャンネルタイルコンポジット当たりのサイクル数は以下の表に要約されている。

【0765】

【表 7 5】

	一定色	画素色
テクスチャ置換	4	4.75
25%背景 + タイルテクスチャ	4	4.75
平均高さアルゴリズム	5	5.75
フィードバック付き平均高さアルゴリズム	5.75	6.5

タイルカラーリング及びコンポジット処理

タイルは、（タイル全体に）一定色を具備するか、又は入力画像から各画素値を取るようにセットされる。これらの両方のケースは、不透明度を拡大縮小するため、テクスチャ処理ステージからのフィードバックがある（シニングペイントと同様）。

【0766】

4 ケースに対するステップは以下のように要約できる：

- －タイルの不透明度値をサブピクセル平行移動する；
- －（テクスチャ貼り付けからのフィードバックが有効である場合）タイルの不透明度を選択的にスケーリングする；
- －画素の色を（一定に、又は画像マップから）決定する；
- －画素を背景画像上でコンポジットする。

【0767】

4 ケースの各々は、関数を実行するために要する時間を最小限に抑えるため、別々に取り扱われる。単一カラーチャンネルのカラーコンポジット処理スタイル当たりの時間は以下の表に要約されている。

【0768】

10

20

30

40

【表 7 6】

タイル貼りカースタイル	テクスチャからのフィードバック無し(画素当たりサイクル数)	テクスチャからのフィードバック付き(画素当たりサイクル数)
タイルは画素毎に一定色	1	2
タイルは入力画像からの画素毎の画素色	1.25	2

一定色

この場合、タイルは、ソフトウェアによって決定された一定色を具備する。ACP31が一つのタイルを設置する間に、ソフトウェアは次のタイルの配置及びカラーリングを決定することができる。

【0769】

タイルの色は、タイル貼りされている画像をスケーリングした画像へのバイリニア補間によって決定することが可能である。スケーリングされた画像は、作成し、画像ピラミッドの所定の位置に保存することが可能であり、タイル演算の全体に対して1回だけ実行すればよい。サイズが128×128である場合、画像は、各次元に関して、128：1で縮小される。

【0770】

フィードバック無し

タイルのテクスチャ処理からのフィードバックが無いとき、タイルは、指定された座標に設置されるだけである。タイル色は、画素の色毎に使用され、コンポジットのための不透明度は、タイルのサブピクセル平行移動処理された不透明度チャンネルから由来する。このケースでは、カラーチャンネル及びテクスチャチャンネルは、タイル貼りパスの間に完全に独立して処理可能である。

【0771】

プロセスの概要は図103に示されている。タイルのサブピクセル平行移動410は、1出力画素当たり1サイクルの平均時間の場合、2個の乗算ALU及び2個のアダーALUを使用して実行することができる。サブピクセル平行移動からの出力は、背景順次読み出しイタレータからの背景画像で一定タイル色412をコンポジット処理411する際に使用されるべきマスクである。

【0772】

コンポジット処理は、1コンポジット当たり1サイクルの平均時間の場合、1個の乗算ALU及び1個のアダーALUを用いて実行できる。したがって、必要条件は、3個の乗算ALU及び3個のアダーALUである。4個の順次イタレータ413-416が必要であり、それらの内容を読み書きするために320nsを要する。サブピクセル並行移動及びコンポジットのために要する1画素当たりの平均サイクル数の場合、バッファを読み書きするための十分な時間がある。

【0773】

フィードバック付き

タイルのテクスチャ処理からのフィードバックがあるとき、タイルは指定された座標に配置される。タイル色は、画素の色毎に使用され、コンポジットのための不透明度は、フィードバックパラメータによってスケーリングされたサブピクセル平行移動処理された不透明度チャンネルから由来する。このようにして、テクスチャ値は、カラー値が貼り付けられる前に計算しなければならない。

【0774】

プロセスの概要は図97に示されている。タイルのサブピクセル平行移動は、1出力画素当たり1サイクルの平均時間の場合、2個の乗算ALU及び2個のアダーALUを使用して実行することができる。サブピクセル平行移動からの出力は、フィードバック順次読み出しイタレータ420から読み出されたフィードバックに応じてスケーリングされるべきマスクである。このフィードバックはスケーラ(1個の乗算ALU)421へ送られる。

【0775】

コンポジット処理426は、1コンポジット当たり1サイクルの平均時間の場合、1個の乗算ALU及び1個のアダーALUを用いて実行できる。したがって、必要条件は、4個の乗算ALU及び4個のアダーALUである。プロセス全体は、平均的に1サイクル内で完了可能であるが、ボトルネックはメモリアクセスである。なぜならば、5個の順次イタレータが必要になるからである。十分なバッファリングが行われる場合、平均時間は1画素当たり1.25サイクルである。

【0776】

入力画像からの色

タイルの画素に着色する一つの方法は、入力画像の画素から色を取り出すことである。ここでも、コンポジット処理には、テクスチャ処理からのフィードバック付きとフィードバック無しの二つの可能性がある。

10

【0777】

フィードバック無し

この場合、タイル色は、入力画像内の対応する画素から簡単にもたらされる。コンポジット処理用の不透明度は、サブピクセルシフトされたタイルの不透明度チャンネルに由来する。

【0778】

プロセスの概要は図105に示されている。タイルのサブピクセル平行移動425は、1出力画素当たり1サイクルの平均時間の場合、2個の乗算ALU及び2個のアダーALUを使用して実行することができる。サブピクセル平行移動からの出力は、(入力画像428から読み出された)タイルの画素色を背景画像429とコンポジット処理426する際に使用されるべきマスクである。このフィードバックは、スケーラ(1個の乗算ALU)431へ送られる。

20

【0779】

コンポジット処理434は、1コンポジット当たり1サイクルの平均時間の場合、1個の乗算ALU及び1個のアダーALUを用いて実行できる。したがって、必要条件は、3個の乗算ALU及び3個のアダーALUである。プロセス全体は平均して1サイクル以内に完了可能であるが、ボトルネックはメモリアクセスである。なぜならば、5個の順次イタレータが必要になるからである。十分なバッファリングが行われる場合、平均時間は1画素当たり1.25サイクルである。

30

【0780】

フィードバック付き

この場合、タイル色は、やはり入力画像内の対応する画素から簡単にもたらされるが、コンポジット処理用の不透明度は、テクスチャ処理パス中に実際に適用されたテクスチャ高さの相対的な量による影響を受ける。このプロセスは図106に示されている。

【0781】

タイルのサブピクセル平行移動431は、1出力画素当たり1サイクルの平均時間の場合、2個の乗算ALU及び2個のアダーALUを使用して実行することができる。サブピクセル平行移動からの出力は、フィードバック順次読み出しイタレータ432から読み出されたフィードバックに応じてスケーリング431されるべきマスクである。

40

【0782】

コンポジット処理は、1コンポジット当たり1サイクルの平均時間の場合、1個の乗算ALU及び1個のアダーALUを用いて実行できる。

【0783】

したがって、必要条件は、4個の乗算ALU及び3個のアダーALUである。プロセス全体は平均して1サイクル以内に完了可能であるが、ボトルネックはメモリアクセスである。なぜならば、6個の順次イタレータが必要になるからである。十分なバッファリングが行われる場合、平均時間は1画素当たり1.5サイクルである。

【0784】

50

タイルテクスチャ処理

各タイルはそのテクスチャチャンネルによって定義される表面テクスチャを有する。テクスチャは、サブピクセルへ移行移動され、次に、出力画像に貼り付けられる。テクスチャコンポジット処理には、

テクスチャ配置と、

25%背景+タイルのテクスチャと、

平均高さアルゴリズムと、

の3種類のスタイルがある。

【0785】

更に、平均高さアルゴリズムはカラーコンポジット処理のためのフィードバックパラメータを節約する。

【0786】

テクスチャコンポジット処理スタイル毎に要する時間は以下の表に要約される。

【0787】

【表77】

タイル貼りカースタイル	画素当たりサイクル数 (テクスチャからのフィードバック無し)	画素当たりサイクル数 (テクスチャからのフィードバック付き)
テクスチャ置換	1	—
25%背景+タイルテクスチャ値	1	—
平均高さアルゴリズム	2	2

テクスチャ置換

このケースでは、タイルからのテクスチャは、図107に示されるように、画像のテクスチャチャンネルを置き換える。タイルのテクスチャのサブピクセル平行移動436は、1出力画素当たり1サイクルの平均時間で、2個の乗算ALU及び2個のアダーALUを使用して実行できる。サブピクセル平行移動からの出力は、順次書き込みイタレータ437へそのまま供給される。

【0788】

テクスチャ置換コンポジット処理に要する時間は1画素当たり1サイクルである。テクスチャ値の100%が常に背景に貼り付けられるので、フィードバックは無い。したがって、チャンネルを特定の順序で処理する必要はない。

【0789】

25%背景+タイルのテクスチャ

この例では、タイルからのテクスチャは既存テクスチャ値の25%に加えられる。新しい値は元の値以上でなければならない。その上、テクスチャチャンネルは8ビットしかないので、新しいテクスチャ値は255でクリップされる。利用されるプロセスは図108に示されている。

【0790】

タイルのテクスチャのサブピクセル平行移動440は、1出力画素当たり1サイクルの平均時間で、2個の乗算ALU及び2個のアダーALUを使用して実行できる。サブピクセル平行移動440からの出力は、アダー441へ供給され、背景テクスチャ値の1/4442に加算される。最小値及び最大値関数444は、サブピクセル平行移動のために使用されない2個のアダーによって設けられ、出力は順次書き込みイタレータ445に書き込まれる。

【0791】

このスタイルのテクスチャコンポジット処理のために要する時間は1画素当たり1サイクルである。テクスチャ値の100%は、（たとえ、255でクリッピングされたとしても）背景に貼り付けられたと考えられるので、フィードバックは無い。したがって、チャンネルを特定の順序で処理する必要はない。

【0792】

平均高さアルゴリズム

このテクスチャ貼り付けアルゴリズムでは、タイルに基づいて平均高さが計算され、各画像の高さは平均高さと比較される。画素の高さが平均よりも小さい場合、ストローク高さが背景高さに加えられる。画素の高さが平均以上である場合、ストローク高さが平均高さに加えられる。このようにして、背景ピークはストロークを細くする。この高さは、背景がストローク貼り付けを0まで細くすることを避けるため、最小限の量で増加するように拘束される（しかし、最小限の量は0になり得る）。高さは、テクスチャチャンネルが8ビット解像度であるため、255で切り取られる。

【0793】

貼り付けられたテクスチャと貼り付けられる予測量との差のフィードバックが存在し得る。フィードバック量は、タイルの色の貼り付けの際に倍率として使用可能である。

【0794】

両方のケースで、平均テクスチャは、ソフトウェアによって提供され、スケーリングされたテクスチャマップに関するバイレベル補間を実行することにより計算される。ソフトウェアは、現在のタイルが貼り付けられている間に、次のタイルの平均テクスチャ高さを決定する。ソフトウェアは、付加する最小限の厚さを与え、この最小限の厚さは、典型的に、タイル貼り付けプロセスの全体で一定である。

【0795】

フィードバック無し

フィードバックが無い場合、テクスチャは、図109に示されるように、背景テクスチャに貼り付けられるだけである。

【0796】

4個の順次イタレータが必要であり、これは、プロセスが1サイクルの間でパイプライン化できるならば、メモリの高速性が遅れないために十分であることを意味する。

【0797】

タイルのテクスチャのサブピクセル平行移動450は、1出力画素当たり1サイクルの平均時間で、2個の乗算ALU及び2個のアダーALUを使用して実行できる。最小値及び最大値関数451、452は、1サイクル以内に全演算を完了するため、別個のアダーALUを必要とする。テクスチャのサブピクセル平行移動によって既に2が使用されているので、1サイクル平均時間のために十分な残り時間が無い。

【0798】

したがって、1画素のテクスチャを処理するための平均時間は2サイクルである。フィードバックが存在せず、コンポジット処理のカラーチャンネルの順序は無関係であることに注意する必要がある。

【0799】

フィードバック付き

これは、概念的にはフィードバック無しの場合と同じであるが、但し、標準的なテクスチャ貼り付けアルゴリズムの処理の他に、実際に貼り付けられたテクスチャの割合を記録しなければならない。この割合は、引き続きタイルの色を背景画像にコンポジット処理するための倍率として使用可能である。フローチャートは図110に示され、以下のルックアップテーブルが使用される。

【0800】

【表78】

ルックアップ	サイズ	詳細
LU ₁	256 エントリー 1 エントリー当たり 16 ビット	1:N テーブルはN (レンジ0-255) によってインデックス付け 結果を固定小数点 0:16 として取扱

10

20

30

40

50

ソフトウェアで設けられた1/Nテーブル460の256個のエントリーの各々は16ビットであり、連続して保持するため16個のキャッシュラインが必要である。

【0801】

タイルのテクスチャのサブピクセル平行移動461は、1出力画素当たり1サイクルの平均時間で、2個の乗算ALU及び2個のアダーALUを使用して実行できる。最小値関数462及び最大値関数463の各々は、1サイクル以内に全演算を完了するため、別個のアダーALUを必要とする。テクスチャのサブピクセル平行移動によって既に2が使用されているので、1サイクル平均時間のために十分な残り時間が無い。

【0802】

したがって、1画素のテクスチャを処理するための平均時間は2サイクルである。フィードバックデータエリア（タイルサイズの画像チャンネル）のために十分な空間を割り付けるべきである。テクスチャは、タイルの色が貼り付けられる前に貼り付けられるべきである。なぜならば、フィードバックがタイルの不透明度をスケーリングするため使用されるからである。

【0803】

CCD画像インターポレータ

ISI83（図3）によってCCDから獲得される像は750×500画素である。像がISIによって撮影されたときのカメラの向きは、画像の上部が「上」に対応するように、画素を0、90、180又は270度回転させるために使用される。全ての画素は、（最大で3というよりも）R、G又はBカラーコンポーネントだけを有するので、これらが回転しているという事実は、画素値を解釈する際に考慮されなければならない。カメラの向きに応じて、2×2画素ブロックの各々は、図111に示されたコンフィギュレーションの何れかを有する。

【0804】

CCD撮影された像を処理のために有用な形式に変換するために、CCD撮影された像に以下の幾つかの処理を行うことが必要である：

－CCD像の低サンプルレートカラーコンポーネントのアップ補間（画素の正しい向きを解釈）；

RGBから内部色空間への変換；

－750×500から1500×1000へ内部空間画像をスケーリング；

－プレーナフォーマットで像を書き込み。

【0805】

像のチャンネル全体は、ワーピングを可能にさせるため、同時に利用できることが必要である。小規模メモリモデル（8MB）の場合、一時的オブジェクトとして全解像度で単一のチャンネルを保持するための空間しかない。このため、色変換は単一のカラーチャンネルへの変換である。このプロセスに関する制約要因は色変換である。なぜならば、色変換は、RGBから内部色空間へのトライリニア補間を含み、このトライリニア補間は、1チャンネル当たり0.026ns（750×500×1画素当たり7サイクル×1サイクル当たり10ns=26250000ns）を要するプロセスだからである。

【0806】

内部色空間画像をスケーリングする前に色変換を実行することが重要である。なぜならば、これは、スケーリングされる画素数（したがって、全体のプロセス時間）を4の倍率で削減するからである。

【0807】

全ての変換に対する必要条件がALUスキームに当てはまる訳ではない。変換は、したがって、以下の二つのフェーズに分解される：

フェーズ1：CCD像の低サンプルレートカラーコンポーネントのアップ補間（画素の正しい向きを解釈）；

RGBから内部色空間への変換；

プレーナフォーマットでの像の書き込み；

フェーズ 2 : 750 × 500 から 1500 × 1000 へ内部空間画像をスケーリング。

【0808】

スケール関数の分離は、小さい色変換後の画像は大きい画像と同時にメモリ内に存在すべきであることを意味する。フェーズ 1 からの出力 (0.5 MB) は、通常、画像ピラミッド (1 MB) のために確保されたメモリエリアに安全に書き込まれる。フェーズ 2 からの出力は、一般的な拡大された CCD 像である。スケーリングの分離によって、更に、スケーリングをアフィン変換によって実現することが可能になり、これは、簡単な 1 : 2 の拡大ではない様々な CCD 解像度を許容する。

【0809】

フェーズ 1 : 低サンプルレートカラーコンポーネントのアップ補間

10

3 カラーコンポーネント (R、G 及び B) の各々は、所与の画素に対して色変換を実行するためアップ補間される必要がある。色変換は 7 サイクルを使用するので、補間を実行するため 1 画素当たり 7 サイクルがある。

【0810】

G の補間は簡単であり、図 112 に示されている。向きに応じて、実際の画素値 G は、奇数ライン上の奇数画素及び偶数ライン上の偶数画素と、偶数ライン上の奇数画素及び奇数ライン上の偶数画素との間で交互に代わる。両方のケースで、リニア補間が要求される全てである。R コンポーネント及び B コンポーネントの補間は、図 113 及び図 113 に示されるようにもっと複雑である。なぜならば、水平方向及び垂直方向において、同図から分かるように、3 行の画素に同時にアクセスすることが要求されるので、3 個の順次読み出しイタレータが必要であり、各順次読み出しイタレータは単一行によってオフセットされる。その上、各行のためのラッチを用いて、同じ行の前の画素へアクセスする。

20

【0811】

したがって、各画素は、CCD からの 1 個のコンポーネントと、他の 2 個のアップ補間されたコンポーネントと、を格納する。一方のコンポーネントがバイリニア補間されているとき、他方はリニア補間されている。補間係数は定数 0.5 であるため、補間は、加算及び 1 ビット右シフトによって (1 サイクル以内に) 計算することが可能であり、係数 0.5 のバイリニア補間は、3 回の加算及び 2 ビット右シフト (3 サイクル) によって計算可能である。必要な総サイクル数は、単一の乗算 ALU を使用した場合 4 である。

【0812】

30

図 115 は、回転 0 の偶数ライン偶数画素 (EL, EP) 及び奇数ライン奇数画素 (OL, EP) のケースを示し、図 116 は、回転 0 の偶数ライン奇数画素 (EL, OP) 及び奇数ライン偶数画素 (OL, OP) のケースを示している。他の回転は、これらの 2 個の表現の少し異なる形式である。

【0813】

色変換

RGB から Lab への色空間変換は、汎用色空間変換関数に関して説明された方法と同じ方法を使用して実現され、1 画素当たり 8 サイクルを要するプロセスである。フェーズ 1 の処理は図 117 を参照して説明される。

【0814】

40

RGB のアップ補間は 4 サイクル (1 個の乗算 ALU) を必要とするが、色空間の変換は、ルックアップ転送時間のため、1 画素当たり 8 サイクル (2 個の乗算 ALU) を必要とする。

【0815】

フェーズ 2

画像スケーリング

このフェーズは、CCD 解像度 (750 × 500) から作業用写真解像度 (1500 × 1000) への画像のアップ補間と関係する。スケーリングは、1 : 2 の縮尺のアフィン変換を実行させることにより行われる。汎用アフィン変換のタイミングは 1 出力画素当たり 2 サイクルであり、本ケースでは、0.03 秒のスケーリング時間の経過を意味する。

50

【0816】

画像照明

画像が処理された後、画像を一つ以上の光源によって照明することができる。光源は以下の何れの型でもよい：

- 1．指向性型－無限の距離があるので、単一方向へ平行光を投じる；
- 2．オムニ型－全方向へ非焦点光を投じる；
- 3．スポット型－特定の目標点へ焦点ビームを投光する。スポット光に関連した円錐及び半影部が存在する。

【0817】

シーンは、反射角を変化させるため関連したバンプマップを有する。周辺光も選択的に照明されたシーンに存在する。

10

【0818】

加速された照明のプロセスでは、単一の光源によって一つの画像チャンネルを照明することに関心がある。多数の光源を、1光源当たり1パスの多数のパスとして、単一の画像チャンネルに適用することが可能である。多数のチャンネルは、バンプマップがある場合も無い場合も、同時に一つずつ処理可能である。

【0819】

ある画素における法線表面ベクトル（N）は、もし存在するならば、バンプマップから計算される。バンプマップが存在しない場合、デフォルト法線ベクトルは、画像面に垂直、即ち、 $N = [0, 0, 1]$ である。

20

【0820】

ビューイングベクトルVは、常に、画像平面に垂直、即ち、 $V = [0, 0, 1]$ である。

【0821】

指向性光源の場合、画素から光源への光源ベクトル（L）は、画像全体中で一定であり、画像全体に対して1回計算される。（有限の距離にある）オムニ光源の場合、光源ベクトルは、画素毎に独立に計算される。

【0822】

周辺光の画素の反射は、 $I_a k_a O_d$ に従って計算される。

【0823】

光源の画素の散乱及び鏡面反射は、Phongモデル：

30

$f_{att} I_p [k_d O_d (N \cdot L) + k_s O_s (R \cdot V)^n]$
に従って計算される。

【0824】

光源が無窮大である場合、光源強度は画像全体に亘って一定である。

【0825】

各光源は1画素当たり3個の寄与度、即ち、

周辺光寄与度

散光寄与度

鏡面寄与度

を有する。

40

【0826】

光源は以下の変数を用いて定義される。

【0827】

【表 7 9】

d_l	光源からの距離
f_{att}	距離による減衰 [$f_{att} = 1/d_l^2$]
R	正規化された反射ベクトル [$R = 2N(NL) - L$]
I_a	周辺光強度
I_p	散光係数
K_a	周辺反射係数
K_d	散光反射係数
K_s	鏡面反射係数
K_{sc}	鏡面色係数
L	正規化された光源ベクトル
N	正規化された表面法線ベクトル
n	鏡面指数
O_d	対象物の散光色 (即ち、画像画素色)
O_s	対象物の鏡面色 ($k_{sc}O_d + (1-k_{sc})I_p$)
V	正規化されたビューイングベクトル [$V = [0,0,1]$]

10

同じ反射係数 (k_a , k_s , k_d) がカラーコンポーネント毎に使用される。

20

【0828】

所与の画素の値は、周辺光寄与と、各光の拡散寄与及び鏡面寄与の合計との和に一致する。

【0829】

照明計算のサブプロセス

散光寄与度を計算するため、様々なその他の計算が必要である。それらの計算の中には、 $1/\Delta X$

N

L

$N \cdot L$

$R \cdot V$

f_{att}

f_{cp}

の計算が含まれる；

サブプロセスは、

周辺光

散光

鏡面

の寄与度を計算するためにも定義される。

30

【0830】

サブプロセスは、光源の全体的な照度を計算するために使用できる。4個の乗算ALUしか存在しないので、特定のタイプの光源に対するマイクロコードは、性能のため適切に混ぜられたサブプロセスを含み得る。

40

【0831】

$1/\sqrt{X}$ の計算

$Vark$ 照明モデルはベクトルを使用する。殆どの場合に、正規化の目的のため、ベクトルの長さの逆数を計算することが重要である。長さの逆数を計算するためには、 $1/([x]$ の平方根) の計算が必要である。

【0832】

論理的に、このプロセスは、図118に示されるような入力及び出力を有するプロセスと

50

して表現可能である。図 1 1 9 を参照すると、この計算は、推定値をルックアップし、次に、以下の関数：

$$V_{n+1} = 1/2 V_n (3 - X V_n^2)$$

を 1 回繰り返すことによって行うことが可能である。

【0 8 3 3】

繰り返しの回数は、要求される精度に応じる。このケースでは、1 6 ビットの精度が必要である。したがって、テーブルは、8 ビットの精度であり、1 回の繰り返しだけが必要である。以下の定数はソフトウェアによってセットされる。

【0 8 3 4】

【表 8 0】

定 数	値
K_1	3

以下のルックアップテーブルが使用される。

【0 8 3 5】

【表 8 1】

ルックアップ	サイズ	詳 細
LU_1	256 エントリー 1 エントリー当たり 8 ビット	$1/\text{SquareRoot}[X]$ の平方根 テーブルは X の最上位 8 ビットによってインデックス付け 結果を固定小数点 0:8 として取扱

N の計算

N は表面法線ベクトルである。バンプマップが存在しない場合、N は一定である。バンプマップが存在する場合、N は画素毎に計算しなければならない。

【0 8 3 6】

バンプマップ無し

バンプマップが無いとき、固定法線 N は、以下の性質、即ち、

$$N = [X_N, Y_N, Z_N] = [0, 0, 1]$$

$$||N|| = 1$$

$$1/||N|| = 1$$

正規化された $N = N$

を有する。

【0 8 3 7】

これらの性質は、具体的に法線ベクトル及び $1/||N||$ を計算する代わりに使用することが可能であり、他の計算を最適化することができる。

【0 8 3 8】

バンプマップ有り

図 1 2 0 に示されるように、バンプマップが存在する場合、N は、X 次元と Y 次元のバンプマップ値を比較することによって計算される。図 1 2 0 は、同じ行及び列における画素に関して画素 P 1 に対する N の計算を示しているが、P 1 自体での値も含まれる。N の計算は、スケール倍率（X 及び Y で同じ倍率）で乗算することにより、解像度に依存しないようにされる。このプロセスは、図 1 2 1 に示されるような入力及び出力（ Z_N は常に 1 である）を有するプロセスとして表現できる。

【0 8 3 9】

Z_N は常に 1 である。 X_N 及び Y_N は未だ正規化されていない（なぜならば、 $Z_N = 1$ である）。N の正規化は、 $N \cdot L$ が計算されるまで遅延されるので、 $1/||N||$ による乗算は、3 回ではなく 1 回である。

【0 8 4 0】

N を計算する実際のプロセスは図 1 2 2 に示されている。

【0841】

以下の定数がソフトウェアによってセットされる。

【0842】

【表82】

定 数	値
K_1	スケールファクタ(Nを解像度から独立にする)

L の計算

10

指向性光

光源が無限遠であるとき、光源は実効的な一定光ベクトル L を有する。L は、

$$L = [X_L, Y_L, Z_L]$$

$$||L|| = 1$$

$$1/||L|| = 1$$

のようにソフトウェアによって正規化され、計算される。

【0843】

これらの性質は、具体的に L 及び $1/||L||$ を計算する代わりに使用することが可能であり、他の計算を最適化することができる。このプロセスは図 123 に示されている。

【0844】

20

オムニ光及びスポット光

光源が無限遠ではないとき、L は現在点 P から光源 P L までのベクトルである。P = [X_P, Y_P, 0]

であるため、L は、

$$L = [X_L, Y_L, Z_L]$$

$$X_L = X_P - X_{PL}$$

$$Y_L = Y_P - Y_{PL}$$

$$Z_L = -Z_{PL}$$

によって与えられる。 $1/||L||$ を乗算することにより、 X_L 、 Y_L 、及び Z_L の各々を正規化する。（後で正規化の際に使用される） $1/||L||$ の計算は、

30

$$V = X_L^2 + Y_L^2 + Z_L^2$$

を計算し、次に、

$$V = 1/2$$

を計算することによって実行される。

【0845】

このケースでは、L の計算は、図 124 に示されるような入力及び出力を有するプロセスとして表現可能である。

【0846】

X_P 及び Y_P は照度が計算されている画像の座標である、 Z_P は常に 0 である。

【0847】

40

L を計算する実際のプロセスは図 125 に示されている。

【0848】

以下の定数がソフトウェアによってセットされる。

【0849】

【表 8 3】

定 数	値
K_1	X_{PL}
K_2	Y_{PL}
K_3	$Z_{PL}^2 (Z_P \text{ が } 0 \text{ のとき})$
K_4	$-Z_{PL}$

N・L の計算

ベクトル N とベクトル L の内積の計算は、

$$X_N X_L + Y_N Y_L + Z_N Z_L$$

によって定義される。

【0850】

バンプマップ無し

バンプマップが無いとき、N は一定 $[0, 0, 1]$ である。したがって、N・L は Z_L に簡約化される。

【0851】

バンプマップ有り

バンプマップが存在するとき、内積を直接計算する必要がある。正規化された N コンポーネントを取り込むのではなく、正規化されていない N と正規化された L の内積を得た後に正規化を行う。L は、（一定であるとき）ソフトウェアによって正規化されるか、又は L 計算プロセスによって正規化される。このプロセスは図 126 に示されている。

【0852】

Z_N は、1 になるように定義されているので、入力として不要である。しかし、その代わりに、結果を正規化するために $1 / ||N||$ が必要とされる。N・L を計算する実際のプロセスは、図 127 に示されている。

【0853】

R・V の計算

R・V は鏡面寄与度の計算への入力として必要である。V = $[0, 0, 1]$ であるため、Z コンポーネントだけが必要である。したがって、R・V は、

$$R \cdot V = 2 Z_N (N \cdot L) - Z_L$$

に簡約化される。

【0854】

加えて、非正規化 $Z_N = 1$ であるため、正規化 $Z_N = 1 / ||N||$ である。

【0855】

バンプマップ無し

最も簡単な実装は、N が一定（即ち、バンプマップ無し）のときである。N 及び V は定数であるため、N・L 及び R・V は、

$$V = [0, 0, 1]$$

$$N = [0, 0, 1]$$

$$L = [X_L, Y_L, Z_L]$$

$$N \cdot L = Z_L$$

$$R \cdot V = 2 Z_N (N \cdot L) - Z_L$$

$$= 2 Z_L - Z_L$$

$$= Z_L$$

として簡単化される。

【0856】

L が一定（指向性光源）であるとき、正規化 Z_L は、R・V が要求されるときに、定数の

10

20

30

40

50

形で与えることができる。Lが変化するとき（オムニ光及びスポット光のとき）、正規化 Z_L はオンザフライ方式で計算しなければならない。それは、L計算プロセスの出力として獲得される。

【0857】

バンブマップ有り

Nが一定ではないとき、 $R \cdot V$ を計算するプロセスは、一般化された式、即ち、

$$R \cdot V = 2 Z_N (N \cdot L) - Z_L$$

を実施するだけである。入力及び出力は図128に示され、実際の実装は図129に示されている。

【0858】

減衰定数の計算

指向性光

光源が無限遠であるとき、光の強度は画像一面で変化しない。したがって、減衰定数 f_{att} は1である。この定数は、無限遠光源の照明計算を最適化するために使用可能である。

【0859】

オムニ光及びスポット光

光源が無限遠ではないとき、光の強度は次式、

$$f_{att} = f_0 + f_1 / d + f_2 / d^2$$

に従って変化する。

【0860】

係数 f_0 、 f_1 及び f_2 を適切にセットすることにより、光強度を一定に、距離に関して線形に、或いは、距離の平方によって減衰させることができる。

【0861】

$d = ||L||$ であるため、 f_{att} の計算は、図130に示されるような入力及び出力を有するプロセスとして表現可能である。

【0862】

f_{att} を計算する実際のプロセスは図131に定義される。

【0863】

以下の定数がソフトウェアによってセットされる。

【0864】

【表84】

定数	値
K_1	F_2
K_2	f_1
K_3	F_0

円錐及び半影ファクタの計算

指向性光及びオムニ光

これらの二つの光源は集光されないので、円錐又は半影を持たない。円錐－半影スケリングファクタ f_{cp} は1である。この定数は、指向性光源及びオムニ光源の照明計算を最適化するために使用できる。

【0865】

スポット光

スポット光は特定の目標点（PT）に集光する。スポット光の強度は、像の特定の点が円錐内にあるか、半影にあるか、又は円錐／半影領域の外側にあるかによって変化する。

【0866】

図132を参照すると、半影一に対する f_{cp} のグラフが示されている。円錐の内部470で f_{cp} は1であり、半影の外部471で f_{cp} は0である。円錐のエッジから半影の

10

20

30

40

50

終わりまで、光強度は 3 次関数 4 7 2 に従って変化する。

【0 8 6 7】

半影 4 7 5 及び円錐 4 7 6 の計算のための様々なベクトルが図 1 3 3 及び図 1 3 4 に示されている。

【0 8 6 8】

図 1 3 4 に示されているように 1 次元に関して像の表面を調べることにより、3 個の角 A、B 及び C が定義される。A は、目標点 4 7 9 と光源 4 7 8 と円錐の終端 4 8 0 の間の角度である。C は、目標点 4 7 9 と光源 4 7 8 と半影の終端 4 8 1 の間の角度である。これらの両方は所与の光源に対して一定である。B は、目標点 4 7 9 と光源 4 7 8 と計算対象一 4 8 2 の間の角度であり、像上で計算される点毎に変化する。

10

【0 8 6 9】

ここで、レンジ A から C を 0 から 1 に正規化し、B が、次式、

$$(B - A) / (C - A)$$

によってその角度レンジの間に収まる距離を見つける。

【0 8 7 0】

そのレンジは 0 から 1 のレンジに丸められ、この値は、 f_{cp} の 3 次近似のためのルックアップとして使用される。

【0 8 7 1】

したがって、 f_{att} の計算は、図 1 3 5 に示されるような入力及び出力を有するプロセスとして表現可能であり、 f_{cp} を計算する実際のプロセスは図 1 3 6 に示され、以下の定数がソフトウェアによってセットされる。

20

【0 8 7 2】

【表 8 5】

定 数	値
K_1	X_{LT}
K_2	Y_{LT}
K_3	Z_{LT}
K_4	A
K_5	$1/(C-A)$ [半影が無いとき最大値]

30

以下のルックアップテーブルが使用される。

【0 8 7 3】

【表 8 6】

ルックアップ	サイズ	詳 細
LU_1	64 エントリー 1 エントリー当たり 16 ビット	Arcos (X) 単位は定数 K_5 及び K_6 に対する単位と同一 テーブルは最上位 6 ビットによってインデックス付け 2 エントリーのリニア補間による結果 タイミングは 2^8 ビット ² エントリー = 4 サイクル
LU_2	64 エントリー 1 エントリー当たり 16 ビット	光応答関数 f_{cp} $F(1) = 0, F(0) = 1$ 、その他は三乗に従う テーブルは 6 ビット (1:5) でインデックス付け 2 エントリーのリニア補間による結果 タイミングは 2^8 ビット = 4 サイクル

40

周辺光寄与の計算

像に当てられる光の数とは無関係に、周辺光寄与が各画素に対して 1 回ずつ実行され、バンプマップには依存しない。

【0 8 7 4】

50

周辺光計算プロセスは図 1 3 1 に示されるような入力及び出力を備えたプロセスとして表現可能である。このプロセスを実装するためには、図 1 3 8 に示されるように入力画像からの各画素 (O_d) を定数値 ($I_a k_a$) によって乗算することが必要であり、以下の定数がソフトウェアによってセットされる。

【0875】

【表 8 7】

定 数	値
K_i	$I_a K_a$

10

散光寄与の計算

表面に当てられた各光は拡散照明を生じる。拡散照明は、次式、

$$\text{散光} = k_d O_d (N \cdot L)$$

によって表される。2 種類の実装形態を考えることができる。

【0876】

実装例 1 — 定数 N 及び L

N 及び L の両方が定数であるとき（指向性光でバンプマップが無いとき）、

$$N \cdot L = Z_L$$

である。したがって、

$$\text{散光} = k_d O_d Z_L$$

である。

【0877】

O_d だけが変数であるため、散光寄与を計算する実際のプロセスは図 1 3 9 に示されるようなものであり、以下の定数がソフトウェアによってセットされる。

【0878】

【表 8 8】

30

定 数	値
K_i	$K_d (N \cdot L) = K_d Z_L$

実装例 2 — 非定数 N 及び L

N 又は L の何れかが非定数であるとき（ビットマップ、又はオムニ光若しくはスポット光からの照明であるとき）、散光計算は、次式、

$$\text{散光} = k_d O_d (N \cdot L)$$

に従って直接実行される。

【0879】

40

散光計算プロセスは図 1 4 0 に示されるような入力を備えたプロセスとして表現可能である。 $N \cdot L$ は、 $N \cdot L$ 計算プロセスを使用して計算するか、又は定数として与えられる。散光寄与を計算する実際のプロセスは図 1 4 1 に示されるようなプロセスであり、以下の定数がソフトウェアによってセットされる。

【0880】

【表 8 9】

定 数	値
K_i	K_d

50

鏡面寄与の計算

表面に当てられた各光は鏡面照明を生ずる。鏡面照明は、次式、

$$\text{鏡面光} = k_s O_s (R \cdot V)^n$$

によって表される。ここで、

$$O_s = k_{sc} O_d + (1 - k_{sc}) I_p$$

である。

【0881】

鏡面光計算プロセスは二つの実装形態がある。

【0882】

実装例1－定数N及びL

10

第1の実装例は、N及びLの両方が定数であるとき（指向性光でバンプマップ無し）である。N、L及びVは一定であるので、N・L及びR・Vも一定である：

$$V = [0, 0, 1]$$

$$N = [0, 0, 1]$$

$$L = [X_L, Y_L, Z_L]$$

$$N \cdot L = Z_L$$

$$R \cdot V = 2 Z_N (N \cdot L) - Z_L$$

$$= 2 Z_L - Z_L$$

20

$$= Z_L$$

このようにして、鏡面光計算は、

$$\text{鏡面光} = k_s O_s Z_L^n$$

$$= k_s Z_L^n (k_{sc} O_d + (1 - k_{sc}) I_p)$$

$$= k_s k_{sc} Z_L^n O_d + (1 - k_{sc}) I_p k_s Z_L^n$$

として簡略化される。

【0883】

鏡面光計算では O_d だけが変数であるため、鏡面光寄与の計算は、図142に示されるような入力及び出力を備えたプロセスとして表現可能であり、鏡面光寄与を計算する実際のプロセスは図143に示され、以下の定数がソフトウェアによってセットされる。

30

【0884】

【表90】

定数	値
K_1	$K_s K_{sc} Z_L^n$
K_2	$(1 - K_{sc}) I_p K_s Z_L^n$

実装例2－非定数N及びL

この実装例は、N又はLの何れかが一定ではないとき（ビットマップ、又はオムニ光若しくはスポット光からの照明であるとき）である。これは、R・Vを与える必要があることを意味し、R・ V^n を計算しなければならない。

40

【0885】

鏡面光計算プロセスは、図144に示されるような入力及び出力を備えたプロセスとして表現可能である。図145は、鏡面光寄与を計算する実際のプロセスを表し、以下の定数がソフトウェアによってセットされる。

【0886】

【表 9 1】

定 数	値
K_1	K_S
K_2	K_{SC}
K_3	$(1-K_{SC}) I_P$

以下のルックアップテーブルが使用される。

【0 8 8 7】

【表 9 2】

ルックアップ	サイズ	詳 細
LU_1	32 エントリー 1 エントリー当たり 16 ビット	X^n テーブルは整数 $R \cdot V$ の最上位 5 ビットによってインデックス付け $R \cdot V$ の小数を使用する 2 エントリーのリニア補間の結果 2 回の乗算による補間 ルックアップからデータを取り出すために要する時間は $2 \cdot 8 \text{ ビット} \cdot 2 \text{ エントリー} = 4 \text{ サイクル}$

10

周辺光が唯一の照明であるとき

周辺光寄与が唯一の光源である場合、このプロセスは非常に簡単である。なぜならば、図 1 4 6 に示されているようなプロセス全体で周辺光を加える必要がないからである。像は、垂直方向に二つの区画に分割し、周辺光ロジックを複製することにより、半分の各々を同時に処理することが可能である（このようにして、全部で 2 個の乗算 A L U 及び 4 個の順次イタレータを使用する）。したがって、タイミングは、周辺光を適用する 1 画素当たり 1 / 2 サイクルである。

20

【0 8 8 8】

典型的な照明のケースは、一つ以上の光によって照らされるシーンである。これらのケースでは、周辺光計算は非常に手軽であるため、周辺光計算は各光源の処理と共に収容される。処理されるべき最初の光は、正確な $I_a k_a$ 設定値を持つべきであり、後続の光は、 $I_a k_a$ の値を 0 にすべきである（多重の周辺光寄与を避けるため）。

30

【0 8 8 9】

周辺光が別個のパスとして（第 1 のパスではなく）処理されるならば、周辺光を現在の計算値に加えることが必要である（同じアドレスへの読み書きが必要である）。プロセスの概要は図 1 4 7 に示されている。

【0 8 9 0】

このプロセスは、3 個の像イタレータと 1 個の乗算 A L U を使用し、平均で 1 画素当たり 1 サイクルを要する。

【0 8 9 1】

無限光源

無限光源のケースでは、像全体で一定の光源強度が得られる。L 及び $f_{a \ t \ t}$ の両方は定数である。

40

【0 8 9 2】

バンプマップ無し

バンプマップが無いとき、一定法線ベクトル $N [0, 0, 1]$ が存在する。照明の複雑さは、N、L 及び $f_{a \ t \ t}$ が一定であることによって著しく軽減される。バンプマップ無しの単一の指向性光を当てるプロセスは図 1 4 7 に示され、以下の定数がソフトウェアによってセットされる。

【0 8 9 3】

【表 9 3】

定 数	値
K_1	I_p

単一の無限光源の場合、図 1 4 8 に示されるような論理演算を実行することが求められ、ここで、 K_1 から K_4 は以下の値をもつ定数である。

【0 8 9 4】

【表 9 4】

定 数	値
K_1	$K_d (N_s L) = K_d L_z$
K_2	K_{sc}
K_3	$K_s (N_s F)^n = K_s H_z^2$
K_4	I_p

このプロセスは、 K_2 、 K_3 及び K_4 が定数であるため簡単化することができる。基本的に（乗算 A L U のうちの 3 個を使用する）鏡面光寄与と散光寄与の計算が複雑であるため、周辺光計算を 4 番目の乗算 A L U として安全に加えることが可能である。処理される第 1 の無限光源は、真の周辺光パラメータ $I_a k_a$ を備え、後続の全ての無限光は、 $I_a k_a$ に 0 をセットすることができる。周辺光計算は実質的に無くなる。

【0 8 9 5】

無限光源が照射される第 1 の光である場合、他の光源による既存の寄与度を取り込む必要はなく、その状況は図 1 4 9 に示され、定数は以下の値をとる。

【0 8 9 6】

【表 9 5】

定 数	値
K_1	$K_d (L_s N) = K_d L_z$
K_4	I_p
K_5	$(1 - K_s (N_s H))^n I_p = (1 - K_s H_z^n) I_p$
K_6	$K_{sc} K_s (N_s H)^n I_p = K_{sc} K_s H_z^n I_p$
K_7	$I_a K_a$

無限光源が照射される第 1 の光ではない場合、先行して処理された光によって生成された既存の寄与度を取り込む必要があり（同じ定数が当てはまる）、状況は図 1 4 8 に示されている。

【0 8 9 7】

第 1 のケースでは、2 個の順次イタレータ 4 9 0、4 9 1 が必要であり、第 2 のケースでは、3 個の順次イタレータ 4 9 0、4 9 1、4 9 2 が必要である（余分のイタレータは先行の光の寄与度を読み出すために必要である）。両方のケースで、バンプマップ無しの無限光源の適用は 1 画素当たり 1 サイクルを要し、オプションとして周辺光の適用が含まれる。

【0 8 9 8】

バンプマップ有り

バンプマップが存在するとき、法線ベクトル N は画素毎に計算することが必要であり、一定光源ベクトル L に適用される。 $1 / ||N||$ は $R \cdot V$ を計算するために使用され、 $R \cdot V$ は鏡面光計算 2 プロセスへの入力として要求される。以下の定数がソフトウェアによ

10

20

30

40

50

ってセットされる。

【0899】

【表96】

定数	値
K_1	X_L
K_2	Y_L
K_3	Z_L
K_4	I_P

10

バンプマップ順次読み出しイタレータ490は、バンプマップの現在ラインを読み出す役割を担う。現在ラインは、Xに関するスロープを決定するため入力として与えられる。バンプマップ順次読み出しイタレータ491、492は、現在ラインの上下のラインを読み出す役割を担う。それらはYに関するスロープを決定するため入力を提供する。

【0900】

オムニ光

オムニ光源のケースでは、光ベクトルL及び減衰定数 f_{att} は像の中で画素毎に変化する。したがって、L及び f_{att} の両方は画素毎に計算しなければならない。

【0901】

バンプマップ無し

バンプマップが無いとき、一定法線ベクトル $N[0, 0, 1]$ が存在する。Lは画素毎に計算しなければならないが、 $N \cdot L$ 及び $R \cdot V$ の両方は、 Z_L に単純化される。バンプマップが無いとき、オムニ光の適用は図149に示されるように計算され、以下の定数がソフトウェアによってセットされる。

20

【0902】

【表97】

定数	値
K_1	X_P
K_2	Y_P
K_3	I_P

30

このアルゴリズムは、オプションとして、先行の光源からの寄与を取り込み、周辺光計算を含む。周辺光は1回だけ取り込めばよい。他の全ての光パスに対して、周辺光計算プロセスにおける適切な定数は0にセットされるべきである。

【0903】

図示されるようなアルゴリズムは全部で19回の乗算／累算を必要とする。ルックアップのために必要な時間は、Lの計算中の1サイクルと、鏡面光寄与計算中の4サイクルである。したがって、5サイクルの処理時間が実現可能な最良のものである。その関数のためにALUを最適にマイクロコード化できない場合には、必要な時間は6サイクルまで増加する。オムニ光を関連したビットマップの無い像へ適用するための速度は1画素当たり6サイクルである。

40

【0904】

バンプマップ有り

オムニ光が関連したバンプマップを備えた像に照射されるとき、N、L、 $N \cdot L$ 及び $R \cdot V$ の計算は全てが必要である。オムニ光を関連したバンプマップを備えた像に適用するプロセスは図150に示され、以下の定数がソフトウェアによってセットされる。

【0905】

【表 9 8】

定 数	値
K_1	X_p
K_2	Y_p
K_3	I_p

このアルゴリズムは、オプションとして、先行の光源からの寄与を取り込み、周辺光計算を含む。周辺光は1回だけ取り込むことが必要である。他の全ての光パスに対し、周辺光計算プロセス内の適切な定数は0にセットされるべきである。

【0906】

図示されるようなアルゴリズムは全部で32回の乗算／累算を必要とする。ルックアップのために要する時間は、LとNの両方の計算中に1サイクルずつと、鏡面光寄与計算のための4サイクルである。しかし、NとLのために必要なルックアップは同じである（したがって、2個のLUが3個のLUを実装する）。8サイクルの処理時間が適切である。その関数のためのALUを最適にマイクロコード化できない場合、必要な時間は9サイクルまで延長される。オムニ光に関連したビットマップ付きの像に適用する速度は1画素当たり9サイクルである。

【0907】

スポット光

スポット光はオムニ光と類似しているが、減衰定数 f_{att} が、目標の周りに効率的に光を集める円錐／半影ファクタ f_{cp} によって変更されている点が相違する。

【0908】

バンプマップ無し

バンプマップが無いとき、一定法線ベクトル $N [0, 0, 1]$ が存在する。Lは画素毎に計算しなければならないが、 $N \cdot L$ 及び $R \cdot V$ の両方は、 Z_L に簡単化される。図151は、像へのスポット光の照射の説明図であり、以下の定数がソフトウェアによってセットされる。

【0909】

【表 9 9】

定 数	値
K_1	X_p
K_2	Y_p
K_3	I_p

このアルゴリズムは、オプションとして、先行の光源からの寄与を取り込み、周辺光計算を含む。周辺光は1回だけ取り込めばよい。他の全ての光パスに対して、周辺光計算プロセスにおける適切な定数は0にセットされるべきである。

【0910】

図示されるようなアルゴリズムは全部で30回の乗算／累算を必要とする。ルックアップのために必要な時間は、Lの計算中の1サイクルと、鏡面光寄与計算中の4サイクルと、円錐／半影計算における2組の4サイクルルックアップである。

【0911】

バンプマップ有り

スポット光に関連したバンプマップを備えた像に照射されるとき、 N 、 L 、 $N \cdot L$ 及び $R \cdot V$ の計算は全てが必要である。単一のスポット光に関連したバンプマップを備えた像に適用するプロセスは図152に示され、以下の定数がソフトウェアによってセットされる。

【0912】

このアルゴリズムは、オプションとして、先行の光源からの寄与を取り込み、周辺光計算

10

20

30

40

50

を含む。周辺光は1回だけ取り込むことが必要である。他の全ての光パスに対し、周辺光計算プロセス内の適切な定数は0にセットされるべきである。図示されるようなアルゴリズムは全部で41回の乗算／累算を必要とする。

【0913】

プリントヘッド44

図153は、論理的に8セグメントにより構成され、各々がページの一部に2層のシアン、マゼンタ、及びイエローを印刷する単一のプリントヘッドの論理的レイアウトの説明図である。

【0914】

印刷用セグメントのローディング

何かを印刷する前に、プリントヘッドの8セグメントの各々は、最終出力画像における以下の相対的な行に対応した6行のデータがロードされなければならない：

行0＝ラインN、イエロー、偶数ドット0、2、4、6、8、...

行1＝ラインN+8、イエロー、奇数ドット1、3、5、7、...

行2＝ラインN+10、マゼンタ、偶数ドット0、2、4、6、8、...

行3＝ラインN+18、マゼンタ、奇数ドット1、3、5、7、...

行4＝ラインN+20、シアン、偶数ドット0、2、4、6、8、...

行5＝ラインN+28、シアン、奇数ドット1、3、5、7、...

各セグメントは、ページの異なる部分にドットを印刷する。各セグメントは、1色の750ドット、即ち、ある行に375個の偶数ドットと、別の行に375個の奇数ドットを印刷する。8個のセグメントは、位置に対応したドットを含む。

【0915】

【表100】

セグメント	最初のドット	最後のドット
0	0	749
1	750	1499
2	1500	2249
3	2250	2999
4	3000	3749
5	3750	4499
6	4500	5249
7	5250	5999

各ドットは、プリントヘッドセグメントに1ビットで表現される。データは、データをセグメントのBit Value（ビット値）ピンに置くことによって、同時に1ビットだけロードされ、Bit Clock（ビットクロック）に従ってそのセグメントのシフトレジスタへクロック供給されなければならない。データは単一のレジスタにロードされるので、ビットをローディングする順序は正しい筈である。データは10MHzの最大レートでプリントヘッドにクロック供給することができる。

【0916】

全てのビットがロードされると、ビットはプリントヘッド出力バッファへ並列に転送され、印刷の準備が完了する。この転送はセグメントのParallel Xfer Clockピン上の1個のパルスによって実行される。

【0917】

印刷制御

電力を節約するため、プリントヘッドの全てを必ずしも同時に印刷しなくてもよい。制御ラインの組は、特定のドットの印刷を可能にさせる。ACPのような外部コントローラは、速度及び／又は電力の必要条件に応じて、同時に印刷されるドット数、並びに、プリントパルスの間隔を変更することが可能である。

【0918】

各セグメントは、5本のNozzleSelect（ノズル選択）ラインを含み、ノズル選択ラインは1行当たり32組のノズルを選択するため復号化される。各行には375個のノズルがあるので、各組には12個のノズルがある。更に、色の偶数行と奇数行に1本ずつの、2本のBankEnable（バンクイネーブル）ラインが設けられる。最後に、各セグメントは、Cカラー、Mカラー及びYカラーの各々に1本ずつの、3本のColorEnable（カラーイネーブル）ラインを含む。あるColorEnableライン上のパルスは、カラーの指定行の指定されたノズルに印刷を行わせる。パルスは、典型的に、2

10

sの間隔である。

【0919】

全てのセグメントが、NozzleSelectラインと、BankEnableラインと、ColorEnableライン（プリントヘッドへ外部から配線されている）の同じ組によって制御される場合、以下の事項が成立する。

【0920】

奇数バンク及び偶数バンクの両方が同時に印刷をする場合（両方のBankEnableビットがセットされている場合）、1セグメント当たり24個のノズル、即ち、全部で192個のノズルが同時に発射し、5.7ワットを消費する。

20

【0921】

奇数バンクと偶数バンクが独立に印刷をする場合、1セグメント当たり12個のノズルだけが、即ち、全部で96個が同時に発射し、2.85ワットを消費する。

【0922】

印刷ヘッドインタフェース62

印刷ヘッドインタフェース62は、ACPをプリントヘッドに接続し、データと適切な信号の両方を外部プリントヘッドへ供給する。プリントヘッドインタフェース62は、VLIWプロセッサ74と、CPU上で実行されるソフトウェアアルゴリズムの両方と協働して動作し、約2秒のうちに写真を印刷する。

30

【0923】

プリントヘッドインタフェースへの入出力の概要は図154に示されている。アドレス及びデータバスがCPUによって使用され、プリントヘッドインタフェースの種々のレジスタのアドレスを指定する。単一のBitClock出力ラインは、プリントヘッド上の8セグメントの全てに繋がる。8本のDataBits（データビット）ラインは、1本ずつが各セグメントに接続され、（BitClockパルスに基づいて）プリントヘッドの8セグメントへ同時にクロックインされる。例えば、同時に、ドット0はセグメント0へ転送され、ドット750はセグメント1へ転送され、ドット1500はセグメント2へ転送され、以下同様である。

40

【0924】

VLIW出力FIFOは、ディザ処理された2レベルC、M及びYの6000×9000解像度印刷画像を、8本のDataBitsラインへ出力するために正しい順序で格納する。ParallelXferClockは、プリントヘッドの8セグメントの各々へ接続されるので、単一パルスによって、全てのセグメントがそれぞれのビットを同時に転送する。最後に、NozzleSelectライン、BankEnableライン及びColorEnableラインが8セグメントの各々へ接続され、プリントヘッドインタフェースは、C、M及びYドロップパルスの間隔、並びに、各パルスで印刷されるドロップの数を制御できるようになる。プリントヘッドインタフェースのレジスタは、0から6µsのパルス間隔、典型的に2µsの間隔を指定することができる。

50

【0925】

画像印刷

画像がA r t c a mのユーザの管理下に置かれる前に、以下の2フェーズ、即ち、

1. 印刷されるべき画像の準備
2. 準備された画像の印刷

を行うことが必要である。

【0926】

画像の準備は1回だけ実行すればよい。画像印刷は必要に応じて何回で実行できる。

【0927】

画像準備

印刷用の画像の準備には、

1. 写真画像の印刷画像への変換
2. プリンタの向きの出力に揃えるために印刷画像（内部色空間）の回転
3. （必要に応じて）圧縮されたチャンネルのアップ補間
4. 内部色空間から特定のプリンタ及びインクに適したC M Y色空間への変換が含まれる。

【0928】

画像準備の最後に、4. 5 M Bの正しく向きを決められた1 0 0 0 × 1 5 0 0 C M Y色空間画像の印刷準備が完了する。

【0929】

写真画像の印刷画像への変換

写真画像を印刷画像へ変換するためには、画像処理を実行するためV a r kスクリプトを実行することが必要である。このスクリプトは、デフォルト画像強調スクリプト、又は現在挿入中のA r t c a r dから取り出されたV a r kスクリプトである。V a r kスクリプトは、C P Uによって実行され、V L I Wベクトルプロセッサによって実行される関数で加速される。

【0930】

印刷画像の回転

メモリ内の画像は、最初、上端が上向きになるように方向を合わされる。これにより、V a r k処理を簡単にすることができる。画像が印刷される前に、画像は、プリントロールの向きと揃えられなければならない。再アライメントは1回だけ実行すればよい。印刷画像の引き続くプリントは、予め適切に回転されているであろう。

【0931】

適用されるべき変換は、ユーザがA r t c a mの「撮像」ボタンを押したときに、C C Dからの捕捉中に適用された変換の単に逆変換である。元の回転が0であるならば、変換は不要である。元の回転が+90度であるならば、印刷前に-90度（270度と同じである）回転する必要がある。この回転を適用するために使用される方法は、V a r k加速されたアフィン変換関数である。アフィン変換エンジンは、各カラーチャンネルを独立に回転させるため呼び出される。カラーチャンネルは元の位置で回転できないことに注意する必要がある。その代わりに、カラーチャンネルは、拡大された単一チャンネル（1. 5 M B）のために先に使用されたスペースを利用することが可能である。

【0932】

図155は、L a b画像の回転の一例の説明図であり、aチャンネル及びbチャンネルは4:1に圧縮されている。Lチャンネルは、不要になったスペース（単一チャンネルエリア）へ回転させられ、次に、aチャンネルは、Lの後に残された空きスペースに回転させられ、最後に、bを回転させることができる。三つのチャンネルを回転するために要する総時間は0.09秒である。これは、最初の画像印刷の前に許容可能な時間の経過である。後続のプリントは、このオーバーヘッドを生じない。

【0933】

アップ補間及び色変換

10

20

30

40

50

L a b 画像は、印刷前に C M Y へ変換しなければならない。L a b 画像の a チャンネル及び b チャンネルが圧縮されているかどうかに応じて、様々な処理が行われる。L a b 画像が圧縮されているならば、a チャンネル及び b チャンネルは色変換が行われる前に伸長されなければならない。L a b 画像が圧縮されていないならば、色変換は唯一の必要なステップである。L a b 画像は、(a チャンネル及び b チャンネルが圧縮されているならば) アップ補間し、C M Y 画像へ変換される。スケール及び色変換を組み合わせた単一の V L I W プロセスを使用することができる。

【0934】

色変換を実行するために使用される方法は、V a r k 加速型色変換関数である。アフィン変換エンジンは、各カラーチャンネルを独立に回転させるため呼び出し可能である。カラーチャンネルはそのままの位置で回転できない。その代わりに、カラーチャンネルは拡大された単一チャンネル (1 . 5 M B) のために先に使用されたスペースを利用することが可能である。

10

【0935】

画像印刷処理

画像印刷処理は、正しい向きに置かれた 1 0 0 0 × 1 5 0 0 C M Y 画像を取り込み、外部プリントヘッドへ送信されるべきデータ及び信号を生成することに関係する。このプロセスは、V L I W プロセス及びプリントヘッドインタフェースと協働する C P U を必要とする。

【0936】

A r t c a m 内の画像の解像度は 1 0 0 0 × 1 5 0 0 である。印刷された画像は、6 0 0 0 × 9 0 0 0 ドットの解像度があり、非常に簡単な関係：1 画素 = 6 × 6 = 3 6 ドットが得られる。図 1 5 6 に示されるように、各ドットは 1 6 . 6 μ m であるため、6 × 6 平方ドットは 1 0 0 平方 μ m である。各ドットは 2 レベルであるため、出力をディザ処理する必要がある。

20

【0937】

画像は約 2 秒で印刷されるべきである。9 0 0 0 行のドットの場合、これは、各行を印刷する間の時間が 2 2 2 μ s の時間であることを意味する。プリントヘッドインタフェースは、この時間内に 6 0 0 0 ドットを生成しなければならない、平均して 1 ドット毎に 3 7 n s である。しかし、各ドットは 3 色により構成されるので、プリントヘッドインタフェースは、約 1 2 n s 内に、即ち、A C P の 1 クロックサイクル (1 0 0 M H z で 1 0 n s) のうちに各カラーコンポーネントを生成しなければならない。一つの V L I W プロセスは、印刷されるべき次の 6 0 0 0 ドットのラインを計算する役割を担う。奇数と偶数の C、M 及び Y ドットは、6 種類の 1 0 0 0 × 1 5 0 0 画像ラインからの入力をディザ処理することによって生成される。第 2 の V L I W プロセスは、先に計算された 6 0 0 0 ドットのラインを取り込み、1 回の転送でプリントヘッドインタフェースからプリントヘッドへ転送されるべき 8 個のセグメントのための 8 ビットのデータを正確に生成する役割がある。

30

【0938】

C P U プロセスは、プリントライン毎に 3 回ずつ (1 カラーコンポーネント当たり 1 回ずつ = 2 秒間に 2 7 0 0 0 回) 第 1 の V L I W プロセス 3 内のレジスタを更新し、第 2 の V L I W プロセス内のレジスタをプリントライン毎に 1 回ずつ (2 秒間に 9 0 0 0 回) 更新する。C P U は、これを行うために、V L I W プロセスよりも 1 ラインだけ先に進んでいる。

40

【0939】

最後に、プリントヘッドインタフェースは、V L I W 出力 F I F O から 8 ビットデータを取り込み、それをそのままプリントヘッドへ出力し、B i t C l o c k 信号を適切に生成する。全てのデータの転送が終わった後、P a r a l l e l X f e r C l o c k 信号が次のプリントラインのデータをロードするため生成される。データのプリントヘッドへの転送と連動して、別個のタイマーが、プリントヘッドインタフェース内部レジスタによって指定された N o z z l e S e l e c t ライン、C o l o r E n a b l e ライン及び B a n

50

kEnableラインを使用して、プリントヘッドの異なるプリントサイクルのための信号を生成する。

【0940】

更に、CPUは、印刷プロセス中にパラレルインタフェースを介して種々のモーター及び裁断機を制御する。

【0941】

C、M及びYドット生成

このプロセスへの入力、印刷のため正しい向きに置かれた1000×1500CMY画像である。この画像は圧縮されていない。図157に示されるように、VLWマイクロコードプログラムは、CMY画像を取り込み、プリントヘッドインタフェースによって要求された、ディザ処理されるC、M及びY画素を生成する。

【0942】

このプロセスは、三つのカラーコンポーネントの各々に対して1回ずつの3回実行される。このプロセスは、並列に実行される二つのサブプロセスを含み、一方は奇数ドットを生成し、他方は偶数ドットを生成する。各サブプロセスは入力画像から1画素を取り込み、3個の出力ドットを生成する（なぜならば、1画素=6出力ドットであり、各サブプロセスは偶数ドットと奇数ドットの何れかに関連しているからである）。1個の出力ドットが1サイクル毎に生成されるが、入力画素は3サイクル毎に1回だけ読み出される。

【0943】

元のディザセルは64×64形のセルであり、各エントリは8ビットである。この元のセルは、奇数セルと偶数セルに分割されるので、各々は64の高さのままであるが、32エントリの幅しかない。偶数ディザセルは、元のディザセル画素0、2、4等を格納し、一方、奇数ディザセルは、元のディザセル画素1、3、5等を格納する。ディザセルはラインを超えて繰り返されるので、2個のディザセルの各々の単一の32バイトラインは、ライン全体の間が必要であり、したがって、完全にキャッシュすることができる。単一のプロセスラインのうちの奇数ライン及び偶数ラインは、8ドットラインだけ離して交互にされるので、奇数ディザセルの8ライン毎に回転させることが好都合である。したがって、奇数ディザセルと偶数ディザセルの両方に同じオフセットを使用することが可能である。この結果として、偶数ディザセルのラインは元のディザセルのラインLの偶数エントリに対応し、奇数ディザセルのラインは元のディザセルのラインL+8の奇数エントリに対応する。

【0944】

このプロセスは、カラーコンポーネントの各々に対して1回ずつの3回実行される。CPUソフトウェアルーチンは、奇数ライン及び偶数ライン用の順次読み出しイタレータがプリントヘッドに対応した正しい画像ラインを確実に指定しなければならない。例えば、1組の18000ドット（3組の6000ドット）を生成するため、

- ・イエロー偶数ドットライン=0、したがって、入力イエロー画像ライン=0/6=0
 - ・イエロー奇数ドットライン=8、したがって、入力イエロー画像ライン=8/6=1
 - ・マゼンタ偶数ドットライン=10、したがって、入力マゼンタ画像ライン=10/6=1
 - ・マゼンタ奇数ドットライン=18、したがって、入力マゼンタ画像ライン=18/6=3
 - ・シアン偶数ドットライン=20、したがって、入力シアン画像ライン=20/6=3
 - ・シアン奇数ドットライン=28、したがって、入力シアン画像ライン=28/6=4
- である。この後続く入力画像ラインの組は、

- ・Y=[0, 1], M=[1, 3], C=[3, 4]
- ・Y=[0, 1], M=[1, 3], C=[3, 4]
- ・Y=[0, 1], M=[2, 3], C=[3, 5]
- ・Y=[0, 1], M=[2, 3], C=[3, 5]
- ・Y=[0, 2], M=[2, 3], C=[4, 5]

である。

【0945】

しかし、ディザセルデータはカラーコンポーネント毎に更新しなくてもよい。3カラーのディザセルは同じになるが、コンポーネント毎に2ドットラインずつオフセットしている。

【0946】

ディザ処理された出力は順次書き込みイタレータに書き込まれ、奇数及び偶数のディザ処理されたドットは2個の別々の出力に書き込まれる。同じ2個の書き込みイタレータが3個の全てのカラーコンポーネントに対して使用されるので、それらは、奇数及び偶数ドットの分割の範囲内で連続的である。

10

【0947】

1組のドットがプリントラインに対して生成される間に、先に生成されたドットの組が、次のセクションで説明されるような第2のVLIWプロセスによって併合される。

【0948】

併合8ビットドット出力生成

このプロセスは、図158に示されるように、1ラインのディザ処理されたドットを取り込み、VLIW出力FIFOを介してプリントヘッドインタフェースへ出力するための8ビットのデータストリームを生成する。このプロセスは、ライン全体が準備されることを必要とする。なぜならば、このプロセスは、ディザ処理されたラインの殆どを同時に準ランダムアクセスする必要があるからである。以下の定数がソフトウェアによってセットされる。

20

【0949】

【表101】

定 数	値
K ₁	375

順次読み出しイタレータは、前に生成されたドットのラインを指定し、イタレータのレジスタは単一カラーコンポーネントへのアクセスを制限するためセットアップされている。後続の画素の間の距離は375であり、あるラインと次のラインとの間の距離は1バイトになるように与えられる。その結果として、8個のエントリーが「ライン」毎に読み出される。単一の「ライン」は、プリントヘッドへロードされる8ビットに対応する。画像の「ライン」総数は375にセットされる。少なくとも8個のキャッシュラインを順次読み出しイタレータに割り当てることにより、完全なキャッシュコヒーレンスが維持される。8ビットをカウントする代わりに、8個のマикроコードステップが暗黙的にカウントを行う。

30

【0950】

生成プロセスは、最初に、偶数ドットから全てのエントリーを読み出し、8個のエントリーを1バイトに合成し、その1バイトをVLIW出力FIFOへ出力する。3000個の全部の偶数ドットが読み出された後、3000個の奇数ドットが読み出され、処理される。ソフトウェアルーチンは、1カラーコンポーネント毎に1回ずつ、即ち、1ライン毎に3回ずつ奇数及び偶数の順次読み出しイタレータのドットのアドレスを更新しなければならない。二つのVLIWプロセスは、全部で8個のALUとVLIW出力FIFOを必要とする。CPUが上述のように二つのプロセスでレジスタを更新できる限り、VLIWプロセッサは、プリンタに追従するために十分な速さでディザ処理された画像ドットを生成し得る。

40

【0951】

データカードリーダー

図159は、読み出すためArtcard9を挿入することができるカードリーダー50

50

0の一形態を示す図である。図158は図159のリーダーの分解斜視図である。カードリーダーはコンピュータシステムに相互連結され、CCD読み出し機構35を含む。カードリーダーは、挿入されたArt card 9を挟むピンチローラー506及び507を含む。一方のローラー、例えば、ローラー506は、二つのローラー506及び507の間のカード9を一樣な速度で前進させるためのArt card モーター37によって駆動される。Art card 9は、一連のLEDライト512の上を通過させられ、一連のLEDライトは、半円形の断面を有する透明なプラスチックモールド514内に収容される。この断面は、例えば、LED512からの光を、カード9がLED512の傍を通過するときに、カード9の表面に集める。その表面から、光は、約480 dpiの解像度に構成された高解像度リニアCCD34へ反射される。Art card 9の表面は、約1600 dpiのレベルまでエンコードされているので、リニアCCD34は、約3倍の乗数でArt card 表面をスーパーサンプルする。Art card 9は、リニアCCD34が1インチ当たり約4800回の読み出しのレートでArt card の移動方向にスーパーサンプルできるような速度で更に駆動される。走査されたArt card のCCDデータはArt card リーダーからACP31へ送られ処理される。光センサにより構成できるセンサ49は、カード13の有無を検出するために機能する。

10

【0952】

CCDリーダーは、下側基板516と、透明な成形されたプラスチックを含む上側基板514と、を有する。二つの基板の間に、半導体製造プロセスによって製作された薄く長いリニアCCDアレイを含むリニアCCDアレイ34が挿入される。

20

【0953】

図160を参照すると、CCDリーダーユニットの一構成例の部分断面斜視側面図が示されている。一連のLED、例えば、LED512は、カード9がCCDリーダー34の表面を通過するときに発光するように動作する。放出された光は、上側基板523の一部分を透過する。この基板は、LED512から放出された光を、カード9上の表面上の点、例えば、点532に集めるように湾曲した周囲を有する部分、例えば、部分529を含む。焦点が合わされた光は、点532から、CCDアレイ34の方へ反射される。誇張されて示された一連のマイクロレンズ、例えば、マイクロレンズ534は、上側基板523の表面に形成される。マイクロレンズ523は、表面全体から受光した光を、CCDアレイ34の感光部に入射する光を検知するCCDリーダー34の表面上の点に対応した下の焦点536に集めるように作用する。

30

【0954】

上記の装置には多数の改良が考えられる。例えば、リニアCCD34上の感知デバイスは千鳥状でもよい。対応したマイクロレンズ34は、千鳥状のCCDセンサに対応するように、光を千鳥状の一連のスポットへ集光させるために相応に形成することができる。

【0955】

読み出しを助けるため、Art card 9のデータ表面エリアは、図38を参照して既に説明したように、チェッカー盤のようなパターンで変調させてもよい。その他の形態の高周波変調も実現可能である。

【0956】

Art card プリンタが記憶装置Art card 上のデータをプリントアウトするものとして提供できることは明らかである。したがって、Art card システムは、Art cam装置の外部に情報を配信する一般的な形態として利用することが可能である。Art card プリンタは、Art card を高品質印刷面にプリントアウトすることが可能であり、多数のArt card を同じ用紙に印刷し、後で分離することができる。Art card 9の第2の面には、引き続き記憶のために、Art card 9に記憶されたファイル等に関する情報を印刷することができる。

40

【0957】

したがって、Art card システムは、CD-ROM、磁気ディスク等の他の形態の記憶装置の代わりに使用するために適した簡単化された形態の記憶装置を可能にさせる。A

50

rt card 9は、大量生産可能であり、そのため、再分配用に実質的に低価格の形態で生産できる。

【0958】

プリントロール

図162には、Art camのプリントロール42及びプリントヘッド部が示されている。用紙／フィルム611は、連続した「ウェブのような」プロセスで、印刷機構15へ送られ、印刷機構15は、更なるピンチローラー616-619、及びプリントヘッド44を含む。

【0959】

ピンチローラー613は、駆動機構（図示せず）に連結され、プリントローラー613の回転時に、フィルム611の形態の「用紙」は、印刷機構615の中へ通され、ピクチャー出力スロット6から出される。回転式裁断機構（図示せず）は、要求された写真サイズで用紙611のロールを切断するために利用される。

【0960】

したがって、プリンタロール42は、写真用にイメージ化されたピクチャーを印刷する印刷機構615に「用紙」611を供給する役割を果たすことが明白である。

【0961】

図163には、プリントロール42の分解斜視図が示されている。プリントロール42は、ピンチローラー612、613の動作の下で出力される出力プリンタ用紙611を含む。

【0962】

次に、図164には、「用紙」フィルムロールが除かれた図163のプリントロール42のより完全な分解斜視図が示されている。プリントロール42は、インク貯蔵セクション620、用紙ロールセクション622、623、及び外側ケーシングセクション626、627からなる三つの主要部品を含む。

【0963】

最初にインク貯蔵セクション620を参照すると、インクリザーバ又はインク供給セクション633が設けられている。印刷用インクは、三つのブラダー型の容器630から632に収容される。プリンタロール42は、フルカラー出力インクを提供するものとする。したがって、第1のインクリザーバ又はブラダー容器630はシアン色インクを収容する。第2のリザーバ631はマゼンタ色インクを収容し、第3のリザーバ632はイエロー色インクを収容する。リザーバ630から632の各々は、容積寸法が異なっているかもしれないが、実質的に同じ容積サイズを有するように設計される。

【0964】

インク貯蔵セクション621、633は、カバー624と共に、プラスチックセクションから製作することが可能であり、ヒートシール、紫外線照射等を用いて一体的にできるように設計される。サイズが等しくされたインクリザーバ630から632の各々は、対応したインクチャンネル639から641に連結され、リザーバ630から632より対応したインク出力ポート635から637へのインクの流れを可能にさせる。インクリザーバ632はインクチャンネル641及び出力ポート637を具備し、インクリザーバ631はインクチャンネル640及び出力ポート636を具備し、インクリザーバ630はインクチャンネル639及び出力ポート7637を具備する。

【0965】

動作中に、インクリザーバ630から632は、対応したインクで充たされ、セクション633はセクション621に接合される。インク貯蔵セクション630から632は、折り畳み可能なブラダーでもよく、インクがインクチャンネル639-641を行き来できるようにさせ、インク出力ポート635から637と流体的に連通される。更に、必要に応じて、空気取り入れ口が設けられ、インクチャンネルリザーバ630から632に関連した圧力を要求通りに維持することができる。

【0966】

10

20

30

40

50

キャップ 6 2 4 は、空気圧取り入れ口からアクセス可能な加圧型キャビティを形成するため、インク貯蔵セクション 6 2 0 に接合することができる。

【0967】

インク貯蔵セクション 6 2 1、6 3 3 及び 6 2 4 は、一体的なユニットとして一つに連結され、プリンタロールセクション 6 2 2、6 2 3 の内側へ挿入できるように設計される。プリンタロールセクション 6 2 2、6 2 3 は、対応したメス型部分（図示せず）と結合するオス型部分 6 4 5 から 6 4 7 を用いてスナップフィットによって一体的に結合されるように設計される。同様に、メス型部分 6 5 4 から 6 5 6 は、対応したオス型部分 6 6 0 から 6 6 2 と結合するように設計される。用紙ロールセクション 6 2 2、6 2 3 は、したがって、一体的にスナップで留められるように設計される。ロール内のフィルム的一端は、二つのセクション 6 2 2、6 2 3 が一つに接合されたとき、二つのセクション 6 2 2、6 2 3 の間で締め付けられる。プリントフィルムは、必要に応じて、プリントロールセクション 6 2 2、6 2 5 上で回転させることができる。

10

【0968】

前述の通り、インク貯蔵セクション 6 2 0、6 2 1、6 3 3、6 2 4 は、用紙ロールセクション 6 2 2、6 2 3 の内側に挿入できるように設計される。プリンタロールセクション 6 2 2、6 2 3 は、オンデマンドでフィルムを与えるため、静止したインク貯蔵セクション 6 2 1、6 3 3 及び 6 2 4 の周りで回転することができる。

【0969】

外側ケーシングセクション 6 2 6 及び 6 2 7 は、更に、プリントロールセクション 6 2 2、6 2 3 の周りで結合できるように設計される。更に、ピンチローラー、例えば、6 1 2、6 1 3 のそれぞれの端は、カバー 6 2 6、6 2 7 内で対応したキャビティ、例えば、6 7 0 に留められるように設計され、ローラー 6 1 3 は、プリントフィルムを供給し、プリントロールから排出するため外部（図示せず）から駆動される。

20

【0970】

最後に、キャビティ 6 7 7 は、プリントロール 4 2 に関連した情報を記憶するためのシリコンチップ集積回路型装置 5 3 を挿入し案内するため、インク貯蔵セクション 6 2 0、6 2 1 に設けることが可能である。

【0971】

図 1 5 5 及び図 1 6 4 に示されるように、プリントロール 4 2 は、Artcam カメラ装置に挿入され、カップリングユニット 6 8 0 と結合するように設計され、カップリングユニット 6 8 0 は、シリコンチップ 5 3 との配線を行うコネクタパッド 6 8 1 を含む。更に、コネクタ 6 8 0 は、インク供給ポート 6 3 5 から 6 3 7 との 4 個の接続用のエンドコネクタを含む。インク供給ポートは、次に、インク供給ライン、例えば、6 8 2 へ接続され、インク供給ラインは、次に、プリントヘッド供給ポート、例えば、6 8 7 へ相互連結され、必要に応じてインクをプリントヘッド 4 4 へ流す。

30

【0972】

ロールを形成するため利用される「メディア」6 1 1 は、適当な画像を上印刷するために設計された様々な材料により構成することが可能である。例えば、不透明な回転可能なプラスチック材料が利用され、透明なプラスチックシートを使用することにより透明性が使用され、メタリックシートフィルムの利用によってメタリック印刷が行われ得る。更に、プリンタロール 4 2 内に布を利用し、布に画像を印刷することも可能であるが、適当な剛性又は適当な裏当てを有する材料だけが利用できることに注意する必要がある。

40

【0973】

印刷メディアがプラスチックであるとき、インクを定着させ、吸収する層でその印刷メディアを覆うことが可能である。更に、幾つかのタイプの印刷メディア、例えば、不透明白色マット、透明フィルム、艶消し透明フィルム、3 次元立体印刷用のレンチキュラーレイフィルム、金属化フィルム、格子又はホログラムのようなエンボス方式の光学的に可変のデバイス、裏面に予め印刷が施されたメディア、及び磁気記録層を含むメディア、を使用できる。金属箔を利用するとき、金属箔は、ポリマーベースをもつことが可能であり、

50

薄い（数ミクロンの）アルミニウム又はその他の金属の蒸着層で覆われ、次に、インクプリンタ機構を介してインクを受容するように適合した透明な保護層で覆われる。

【0974】

使用されるプリントロール42は、カメラ装置の内部に挿入され、要求に応じて画像の印刷用のインク及び用紙を提供できるように設計される。インク出力ポート635から637は、カメラ装置内の対応したポートと適合する。ピンチローラー672、673は、カメラ装置の制御下で用紙をカメラ装置へ供給できるように作動される。

【0975】

図164に示されるように、実装されたシリコンチップ53は、プリントロール42の一方の端に挿入される。図165では、認証チップ53がより詳細に示され、この認証チップ53は、チップ53からチップが挿入されている対応したカメラへ詳細を通信する4本の通信リード線680から683を含む。

10

【0976】

図165を参照すると、チップは、小型集積回路687をエポキシに入れ、ボンディングリード線、例えば、688を外部通信リード線680から683へ繋ぐことにより、単独で作成され得る。集積チップ687は、約400平方ミクロンであり、スクライブ境界は100ミクロンである。続いて、チップは、プリントロール42のキャビティの適切な表面に貼り付けられる。図165の装置の分解図である図166には、ボンディングパッド681、682に相互連結された集積回路67が示されている。

20

【0977】

図164Aから164Eにおいて、参照番号1100は、全体的にプリントカートリッジ1100を示す。プリントカートリッジ1100は、本発明によれば、インクカートリッジ1102を含む。

【0978】

プリントカートリッジ1100はハウジング1104を含む。図2により詳細に示されているように、ハウジング1104は、上側成形部品1106及び下側成形部品1108によって画成される。成形部品1106及び1108は、クリップ1110によって挟まれる。ハウジング1104はラベル1112によって被われ、ラベル1112はカートリッジ1100に魅力的な外観を与える。ラベル1112は、ユーザがカートリッジ1100を使えるようにするための情報を搭載する。

30

【0979】

ハウジング1104はチャンバー1114を画成し、チャンバー内にインクカートリッジ1102が収まる。インクカートリッジ1102は、ハウジング1104のチャンバー1114に固定して担持される。

【0980】

フォーマーに巻き付けられたフィルム／メディア1118のロール1126を含むプリントメディア1116の補給品はハウジング1104のチャンバー1114に収容される。フォーマー1120は、インクカートリッジ1102に摺動自在に収容され、インクカートリッジに対して回転自在である。

【0981】

図164Bに示されるように、上側成形部品1106及び下側成形部品1108は一体としてクリップされ、出口スロット1122が画成され、用紙1118の舌状部は出口スロットを通して排出される。

40

【0982】

カートリッジ1100はローラー組立体1124を含み、ローラー組立体1124は、用紙1118がロール1126から供給されるときに用紙1118のカールを取り除くために役立ち、スロット1122を介して用紙1118を押し出すために役立つ。ローラー組立体1124は、駆動ローラー1128と、2個の従動ローラー1130と、を含む。従動ローラー1130は、リブ1132に回転自在に支持され、リブ1132は、ハウジング1104の下側成形部品1108のフロア1134から盛り上がっている。ローラー1

50

130は、駆動ローラー1128と一体となって、用紙1118がハウジング1104から排出されるときに、用紙1118の速度及び位置を制御するため、用紙1118に正のトラクションを与える。ローラー1130は、ポリスチレンのような適当な合成プラスチック材料の射出成形部品である。この点に関して、上側成形部品1106及び下側成形部品1108は、同様に、ポリスチレンのような適当な合成プラスチック材料の射出成形部品である。

【0983】

駆動ローラー1128は、ハウジング1104の上側成形部品1106及び下側成形部品1108の各々の側壁に画成された合わせリセス1138及び1140の間に回転自在に係留されている駆動シャフト1136を含む。駆動ローラー1128の反対側の端1142は、ハウジング1140の上側成形部品1106と下側成形部品1108に適当な構成（図示せず）で回転自在に保持されている。

10

【0984】

駆動ローラー1128は、ツーショット射出成形部品であり、高衝撃ポリスチレン製のシャフト1136を含み、シャフト1136の上に、エラストマ又はゴムローラー部分1144の形をした軸受手段が成形されている。これらの部分1144は、用紙1118を確実に係止し、用紙1118がカートリッジ1100から供給されるときに、用紙1118の滑りを防止する。

【0985】

ハウジング1104から突起するローラー1128の端は、十字形構造体1146（図164A）の形をした係合体を有し、この十字形構造体は、プリントカートリッジ1100が取り付けられるカメラのような装置のプリントヘッド組立体の歯車付き駆動インタフェース（図示せず）と結合する。この構造体は、用紙1118がプリントヘッドに供給される速度を、プリントヘッドによる印刷と確実に同期させるので、インクが用紙1118上で正確に位置合わせされることを保証する。

20

【0986】

インクカートリッジ1102は、直円柱押出成形部品の形をした容器1148を含む。この容器1148は、ポリスチレンのような適当な合成プラスチック材料から押出成形される。

【0987】

本発明の好適な実施形態において、プリントカートリッジ1100が共に使用されるプリントヘッドは、多色プリントヘッドである。したがって、容器1148は、複数の、より詳細には、4個のコンパートメント、即ち、リザーバ1150に分割される。各リザーバ1150は、異なる色、又は異なるタイプのインクを収容する。一実施形態では、リザーバ1150に収容されるインクは、シアンインク、マゼンタインク、イエローインク、及びブラックインクである。本発明の他の実施形態では、3種類の有色インクであるシアンインク、マゼンタインク及びイエローインクがリザーバ1150のうちの3個に収容され、4番目のリザーバ1150は、赤外線スペクトルだけで可視化されるインクを収容する。

30

【0988】

図164C及び164Dに明瞭に示されているように、容器1148の一端は、エンドキャップ1152によって閉じられる。エンドキャップ1152は複数の開口1154が画成されている。開口1154は各リザーバ1150と関連付けられているので、エンドキャップ1152が設けられた容器1148の端でリザーバ1150内は大気圧で維持される。

40

【0989】

シール構造体1156は、エンドキャップ1152が設けられている端側で容器1148に設けられる。シール構造体1156は、各リザーバ1150に摺動自在に受容されたゲル状材料からなる四分円形ペレット1158を含む。ペレット1158のゲル状材料は、熱硬化性ゴムと炭化水素からなる化合物である。炭化水素は白色鉱物油である。熱硬化性

50

ゴムは、ペレット 1 1 5 8 が通常の動作温度でその形状を維持し、同時に、ペレット 1 1 5 8 がその関連したリザーバ 1 1 5 0 内で摺動できるように、鉱物油に十分な剛性を与える共重合体である。適当な熱硬化性ゴムは、シェル石油会社によって「K a r t o n」（登録商標）の名前で販売されている熱硬化性ゴムである。共重合体は、ゲルのような粘度を各ペレット 1 1 5 8 に与えるために十分な量でその化合物内に存在する。典型的に、共重合体は、使用されるタイプに応じて、約 3 % から 2 0 % の重量パーセントの量で存在するであろう。

【0990】

使用中に、この化合物は加熱され、その結果として流体になる。各リザーバ 1 1 5 0 に特定のタイプのインクが詰められると、溶融状態の化合物は各リザーバ 1 1 5 0 に注入され、化合物はペレット 1 1 5 8 を形成するため固まる。ペレット 1 1 5 8 の裏側の大気圧、即ち、エンドキャップ 1 1 5 2 と向かい合うペレットの端における大気圧は、インクがリザーバ 1 1 5 0 から引き出されるときに、自動注油式であるペレット 1 1 4 8 が容器 1 1 4 8 の反対側へ向かって滑ることを保証する。ペレット 1 1 5 8 は、逆さにされた容器からインクが出て空になることを止め、リザーバ 1 1 5 0 内のインクが汚れることを妨げ、リザーバ 1 1 5 0 内のインクが乾燥することを妨げる。更に、ペレット 1 1 5 8 は、リザーバ 1 1 5 0 からのインクの漏れを防止するため疎水性である。

【0991】

容器 1 1 4 8 の反対側の端は、インク鋳成形部品 1 1 6 0 によって閉じられる。成形部品 1 1 6 0 に支持されたバッフル 1 1 6 2 は、エラストマーシール成形部品 1 1 6 4 を受容する。疎水性であるエラストマーシール成形部品 1 1 6 4 は、内部にシーリング幕 1 1 6 6 が画成されている。各シーリング幕 1 1 6 6 にはスリット 1 1 6 8 が設けられているので、プリントヘッド組立体からの合わせピン（図示せず）は、スリット 1 1 6 8 を通して挿入可能であり、容器 1 1 4 8 のリザーバ 1 1 5 0 と流体的に連通される。中空突起 1 1 7 0 は、インク鋳成形部品 1 1 6 0 の反対側から突起する。各突起 1 1 7 0 は、関連したリザーバ 1 1 5 0 とぴったり合うように成形され、インク鋳成形部品を容器 1 1 4 8 の端に位置決めする。

【0992】

図 1 6 4 C を参照すると、インク鋳成形部品 1 1 6 0 は、キャリア又は隠し板成形部品 1 1 7 2 によって正しい位置に維持される。隠し板成形部品 1 1 7 2 には、四つ葉のクローバー形状の窓 1 1 7 4 が画成され、この窓 1 1 7 4 を介して、エラストマーシール成形部品 1 1 6 4 へアクセスすることができる。隠し板成形部品 1 1 7 2 は、ハウジング 1 1 0 4 の上側成形部品 1 1 0 6 と下側成形部品 1 1 0 8 との間に係留される。隠し板成形部品 1 1 7 4 と、ハウジング 1 1 0 4 の上側成形部品 1 1 0 6 及び下側成形部品 1 1 0 8 のそれぞれの内面から延びるウェブ 1 1 7 6 及び 1 1 7 8 は、コンパートメント 1 1 8 0 を画成する。エアーフィルタ 1 1 8 2 はコンパートメント 1 1 8 0 に収容され、端部成形部品 1 1 7 4 によって正しい位置に保たれる。エアーフィルタ 1 1 8 2 はプリントヘッド組立体と協働する。エアーは、プリントヘッド組立体のノズルガードの至る所へ吹き付けられ、ノズルガードのクリーニングを行う。このエアーは、隠し板 1 1 7 2 内の入口側開口 1 1 8 4 に収容されたピン（図示せず）を用いてエアーフィルタ 1 1 8 2 を通して引き出すことによって濾過される。

【0993】

エアーフィルタ 1 1 8 2 は、図 1 6 4 E に更に詳細に示されている。エアーフィルタ 1 1 8 2 はフィルタ媒体 1 1 9 2 を含む。フィルタ媒体 1 1 9 2 は、合成ファイバに基づくものであり、フィルタリング目的のため利用可能な表面積を増加させるため溝付きの形で配置される。紙ベースのフィルタ媒体 1 1 9 2 の代わりに、他の繊維性芯材を使用してもよい。

【0994】

フィルタ媒体 1 1 9 2 は密閉容器 1 1 9 4 に収容される。密閉容器 1 1 9 4 は、底成形部品 1 1 9 6 及び蓋 1 1 9 8 を含む。ハウジング 1 1 0 4 のコンパートメント 1 1 8 0 に収

容するために、密閉容器 1 1 9 4 は、部分的に環状、即ち、馬蹄形にされる。このようにして、密閉容器 1 1 9 4 は、一对の対向端 1 2 0 0 を有する。吸気開口 1 2 0 2 が各端 1 2 0 0 に画成される。

【0995】

排気開口 1 2 0 4 は蓋 1 1 9 8 に画成される。排気開口 1 2 0 4 は、最初、フィルム、又は膜 1 2 0 6 によって閉鎖されている。フィルタ 1 1 8 2 がコンパートメント 1 1 8 0 の正しい位置に取り付けられたとき、排気開口 1 2 0 4 は隠し板成形部品 1 1 7 2 の開口 1 1 8 4 と位置合わせされる。プリントヘッド組立体からのピンがフィルム 1 2 0 6 を貫通し、エアーがノズルガードとプリントヘッド組立体のプリントヘッドに吹き付けられる前に、エアーフィルタ 1 1 8 2 を介して雰囲気からエアーを引き込む。

10

【0996】

底成形部品 1 1 9 4 は、密閉容器 1 1 9 4 の正しい位置にフィルタ媒体 1 1 9 2 を設置するための位置決め構造体 1 2 0 8 及び 1 2 1 0 を含む。位置決め構造体 1 2 0 8 は複数のピンの形をなし、一方、位置決め構造体 1 2 1 0 は、フィルタ媒体 1 1 9 2 の端 1 2 1 4 を係止するリブの形をしている。

【0997】

フィルタ媒体 1 1 9 2 が底成形部品 1 1 9 6 の正しい位置に設置された後、蓋 1 1 9 8 は、蓋 1 1 9 8 で底成形部品 1 1 9 6 を密閉するため超音波溶接、又は類似した方法によって、底成形部品 1 1 9 6 に固定される。

【0998】

プリントカートリッジ 1 1 0 0 が組み立てられたとき、膜又はフィルム 1 1 8 6 は、窓 1 1 7 4 を閉鎖するために隠れ板成形部品 1 1 7 2 の外側端に被せられる。この膜又はフィルム 1 1 8 6 は、使用するために、ピンによって孔を開けられ、又は破裂せられる。フィルム 1 1 8 6 は、破片がインクリザーバ 1 1 5 0 に侵入することを防止する。

20

【0999】

認証チップ 1 1 8 8 の形の認証手段は、隠れ板成形部品 1 1 7 2 の開口 1 1 9 0 に收容される。認証チップ 1 1 8 8 は、プリントカートリッジ 1 1 0 0 が装置のプリントヘッド組立体と互換性があり、準拠していることを保証するため、プリントヘッド組立体 1 1 8 8 による問い合わせを受ける。

【1000】

認証チップ

30

認証チップ 5 3

好適な実施形態の認証チップ 5 3 は、正しく製造されたプリントロールだけがカメラシステムで利用されることを保証する責任がある。認証チップ 5 3 は、プリントロールシステムに限定されない任意の消耗品に関して利用されるときに一般的に有用である技術を利用する。消耗品を必要とする（トナーカートリッジを必要とするレーザープリンタのような）他のシステムの製造者は、消耗品を認証する問題で苦勞し、成功のレベルは様々である。その殆どは、専用化されたパッケージングに頼っている。しかし、これは、家庭での詰め替え作業や複製品製造を抑えない。コピー作成を防止することは、不完全に製造された代用消耗品が基本システムを損傷することを防ぐために重要である。例えば、濾過が不十分なインクは、インクジェットプリンタのプリントノズルを詰まらせ、消費者はシステム製造者に責任を負わせ、許可されていない消耗品の使用を認めない。

40

【1001】

認証の問題を解決するために、認証チップ 5 3 は、認証コードと、コピー防止のために特別に設計された回路と、を含む。このチップは、標準的なフラッシュメモリ製造プロセスを使用して製造され、インク及びトナーカートリッジのような消耗品に組み込むことができる程度に低価格である。一旦プログラミングされると、認証チップは、後述するように、NSA持ち出し（エクスポート）ガイドラインに準拠する。認証は、非常に大規模であり、且つ常に成長している分野である。以下の説明は、消耗品の認証だけに関連している。

50

【1002】

記号名

以下の記号名が本実施形態の説明を通じて使用される。

【1003】

【表102】

記号名	説 明
$F[X]$	1 個のパラメータ X をとる関数 F
$F[X, Y]$	2 個のパラメータ X 及び Y をとる関数 F
$X Y$	X と Y の連結
$X \wedge Y$	ビット論理積 $X \text{ AND } Y$
$X \vee Y$	ビット論理和 $X \text{ OR } Y$ (包括的論理和)
$X \oplus Y$	ビット排他的論理和 $X \text{ XOR } Y$ (排他的論理和)
$\neg X$	ビット否定 $\text{NOT } X$ (補数)
$X \leftarrow Y$	X に値 Y を割り当て
$X \leftarrow \{Y, Z\}$	X への割り当て入力のドメインは Y 及び Z
$X = Y$	X は Y と等価
$X \neq Y$	X は Y と等価ではない
$\Downarrow X$	X を 1 ずつデクリメント (フロア 0)
$\Box X$	X を 1 ずつインクリメント (レジスタ長に基づくラップ付き)
Erase X	フラッシュメモリレジスタ X の消去
SetBits [X, Y]	Y に基づくフラッシュメモリレジスタ X のビットシフト
$Z \leftarrow \text{ShiftRight } [X, Y]$	レジスタ X を右へ 1 ビットシフトし、 Y からの入力ビットを取得し、出力ビットを Z に入れる

基本用語

M で示されたメッセージは平文である。 M を、 M の実体が隠された暗号文 C に変換するプロセスは、暗号化と呼ばれる。 C を M へ逆変換するプロセスは、復号化と呼ばれる。暗号化関数を E で表し、復号化関数を D で表すことにより、以下の恒等式：

$$E[M] = C$$

$$D[C] = M$$

が得られる。したがって、以下の恒等式：

$$D[E[M]] = M$$

が成り立つ。

【1004】

対称暗号化方式

対称暗号化アルゴリズムとは：

暗号化関数 E が鍵 K_1 に依存する；復号化関数 D が鍵 K_2 に依存する； K_2 は K_1 から導出可能である； K_1 は K_2 から導出可能である；

というアルゴリズムである。

【1005】

殆どの対称アルゴリズムにおいて、 K_1 は通常 K_2 と等しい。しかし、 K_1 が K_2 と等しくない場合であっても、一方の鍵がもう一方の鍵から導出可能であるならば、数学的な定義のためには単一の鍵があれば十分である。したがって、

$$E_K[M] = C$$

$$D_K[C] = M$$

と表せる。

【1006】

誰でも知っている教科書から洗練された現代のアルゴリズムまでの非常に多数の対称アルゴリズムが存在する。それらの殆どは、現在の暗号解読技術がKを導出し得る程度で攻撃に成功し得るという点において不確かである。特定の対称アルゴリズムの安全性は、通常、アルゴリズムの強度、及び鍵の長さからなる二つの事項の関数である。以下のアルゴリズムは、認証チップに利用するために適した特徴を含んでいる：

DES；

ブローフィッシュ (Blowfish)；

RC5；

IDEA。

10

【1007】

DES

DES (データ暗号化規格) は米国及び国際規格であり、暗号化と復号化に同じ鍵が使用される。鍵の長さは56ビットである。これは、ハードウェアとソフトウェアで実装されるが、当初の設計はハードウェア専用であった。DESで使用される当初のアルゴリズムは、米国特許第3962539号に記載されている。トリプルDESの変形と称されるDESの変形はより安全性が高いが、3個の鍵： K_1 、 K_2 及び K_3 を必要とする。これらの鍵は、

$$E_{K_3} [D_{K_2} [E_{K_1} [M]]] = C$$

$$D_{K_3} [E_{K_2} [D_{K_1} [C]]] = M$$

20

という形で使用される。

【1008】

トリプルDESの主な利点は、既存のDESの実装を使用して単一鍵のDESよりも安全性を高くすることができる点である。特に、トリプルDESは、112ビットの鍵の長さと等価的な保護を与える。トリプルDESは、単純に期待されるような168ビット鍵 (3×56) と等価的な保護を与えるものではない。トリプルDESの復号化及び／又は符号化を実行する機器は米国から持ち出すことができない。

【1009】

ブローフィッシュ (Blowfish)

Blowfishは、1994年にSchneierによって最初に提案された対称ブロック暗号方式である。これは、32ビットから448ビットまでの可変長鍵を使用する。更に、これはDESよりもかなり高速である。Blowfishアルゴリズムは、鍵拡張部と、データ暗号化部の二つの部分により構成される。鍵拡張は、最大で448ビットの鍵を、全部で4168バイトの複数のサブ鍵配列へ変換する。データ暗号化は、16ラウンドのFeistelネットワークによって行われる。全ての演算は、32ビット語上でのXORと加算であり、1ラウンド当たりに4回のインデックス配列ルックアップを含む。復号化は、サブ鍵が逆順に使用されることを除いて暗号化と同じである点に注意する必要がある。したがって、実装の複雑さは、このような対称性をもたない他のアルゴリズムよりも低減される。

30

【1010】

40

RC5

1995年にRon Rivestによって設計されたRC5は、ブロックサイズ、鍵のサイズ、及びラウンド回数が可変である。しかし、典型的に、RC5は、64ビットのブロックサイズと128ビットの鍵を使用する。RC5アルゴリズムは、鍵拡張部と、データ暗号化部の二つの部分により構成される。鍵拡張は、鍵を $2r + 2$ 個のサブ鍵 (ここで、 r = ラウンド回数である。) に変換し、各サブ鍵は w ビットである。64ビットのブロックサイズで、16回のラウンドの場合 ($w = 32$, $r = 16$)、サブ鍵配列は全部で136バイトである。データ復号化は、モジュロ 2^w の加算と、XORと、ビット回転と、を使用する。

【1011】

50

I D E A

1990年にL a i及びM a s s e yによって開発されたI D E A暗号方式の最初の具体化された形はP E Sと呼ばれる。1991年にB i h a m及びS h a m i rによって差分解読法が発見された後、このアルゴリズムは強化され、その結果は1992年にI D E Aとして発表された。I D E Aは、64ビットの平文ブロックを操作するために128ビット鍵を使用する。暗号化と復号化のために同じアルゴリズムが使用される。このアルゴリズムは、一般的に、現在利用可能な最も安全なブロックアルゴリズムであると考えられている。これは、1993年に発行された米国特許第5214703号に記載されている。

【1012】

非対称暗号化方式

他に使用できるアルゴリズムは非対称アルゴリズムである。非対称暗号化アルゴリズムとは：

暗号化関数Eが鍵 K_1 に依存する；

復号化関数Dが鍵 K_2 に依存する；

K_2 は合理的な時間内に K_1 から導出不能である；

K_1 は合理的な時間内に K_2 から導出不能である；

というアルゴリズムである。

【1013】

したがって、

$$E_{K_1}[M] = C$$

$$D_{K_2}[C] = M$$

である。

【1014】

これらのアルゴリズムは、一方の鍵 K_1 が公開されるので、公開鍵とも称される。したがって、誰でも(K_1 を使用して)メッセージを暗号化し得るが、対応した復号鍵(K_2)を持っている人だけがそのメッセージを復号化して読むことができる。殆どの場合に、以下の恒等式：

$$E_{K_2}[M] = C$$

$$D_{K_1}[C] = M$$

が成立する。

【1015】

この恒等式は非常に重要である。なぜならば、公開鍵 K_1 を所持する人は誰でもMを見ることができ、そのMは K_2 の所有者からのものであることが分かるからである。他の人はCを生成することができない。なぜならば、Cを生成し得ることは、 K_2 を知っていることを意味するからである。合理的な時間内に K_2 から K_1 を導出できないこと、並びに、その逆に K_1 から K_2 を導出できないことは、当然、合理的な時間という概念によって曖昧になる。再々実証されているように、長時間を要することが予想される計算は、より高速化されたコンピュータ、新しいアルゴリズム等の導入によって実現可能になる。非対称アルゴリズムの安全性は、大きい数(より具体的には、二つの大きい素数の積である大きい数)を素因数分解すること、及び有限において離散対数関数を計算することの困難さの二つの問題のうちの一方の困難さに基づいている。大きい数の素因数分解は、現代の数学の理解においては難問であることが推測される。しかし、素因数分解は予測されるよりも速く容易になっているということが問題である。R o n

R i v e s tは、1977年に、125桁の素因数分解には40000兆年かかると言っていた。1994年に129桁の素因数分解が行われた。S c h n e i e rによれば、1980年代における512ビットの数から得られた安全性のレベルを現在得るためには、1024ビットの数が必要であるとされている。その鍵が数年続くとすると、1024ビットでも十分ではない。R i v e s tは1990年に鍵の長さの評価を改め、2005年まで高い安全性を維持するためには1628ビットを提唱し、2015年まで持ちこたえる高い安全性のためには1884ビットを提唱している。一方、S c h n e i e rは、2

10

20

30

40

50

015年まで企業及び政府からの保護を得るためには2048ビットが必要であることを示唆している。

【1016】

多数の公開鍵暗号化アルゴリズムが存在する。その殆どは実際には実装不可能であり、その多くは、所与のMに対して非常に大きいCを生成するか、又は巨大な鍵を必要とする。更に、その他のものは、たとえ安全であるとしても、非常に低速であるため数年の間には実施できない。このため、多数の公開鍵システムは、ハイブリッド方式であり、公開鍵の仕組みは対称セッション鍵を送信するために使用され、セッション鍵が実際のメッセージのために使用される。全てのアルゴリズムは、鍵選択の点で問題がある。乱数はどうしても安全性が不十分である。大きい素数p及びqは慎重に選ぶことが必要であり、ある種の組み合わせは簡単に素因数分解できるという弱点がある（一部の弱い鍵は検査することができる）。しかし、それにもかかわらず、鍵選択は、例えば、単に、1024ビットをランダムに選択するというような単純なことではない。その結果として、鍵選択プロセスも安全でなければならない。

10

【1017】

公開の精査の下で使用中の実際的なアルゴリズムの中で、以下のアルゴリズム：

R S A ；

D S A ；

E I G a m a l ；

は利用するために適している。

20

【1018】

R S A

R i v e s t、S h a m i r 及び A d l e m a n の名前にちなんで命名されたR S A 暗号システムは、最も広く使用されている公開鍵暗号システムであり、世界中の多くで事実上の標準である。R S A の安全性は、2 個の素数（p 及び q）の積である大きい数の素因数分解の困難さに依存していると考えられる。p 及び q の生成には多数の制約がある。それらは、両方共に大きいことが必要であり、ビット数が同じであり、しかも、互いに近似してはならない（そうでなければ、 $p q$ は $\sqrt{p q}$ と近似する）。その上、多くの場合に、p 及び q は強い素数でなければならないことが指摘されている。R S A アルゴリズムの特許は1983年に発行された（米国特許第4405829号）。

30

【1019】

D S A

D S A（デジタル署名規格）は、デジタル署名標準（D S S）の一部として設計されたアルゴリズムである。規定されているとおり、このアルゴリズムは一般的な暗号化には使用できない。その上、R S A に対して、D S A は署名検証の際に10から40倍速度が遅い。D S A は、S H A - 1 ハッシュアルゴリズム（以下の一方向関数の定義を参照せよ。）を明示的に使用する。D S A 鍵生成は、q が p - 1 を割り切るような2 個の素数 p 及び q の検出に依拠している。S c h n e i e r によれば、1024ビットのp 値が長期間のD S A 安全性のために必要である。しかし、D S A 標準は、p の値が1024ビットよりも大きくなることを認めていない（p は、更に、64ビットの倍数でなければならない）。米国政府はD S A アルゴリズムを所有し、少なくとも一つの関連特許（1993年に発行された米国特許第5231688号）を保持する。

40

【1020】

E I G a m a l

E I G a m a l 方式は、暗号化とデジタル署名の両方に使用される。その安全性は有界で離散対数関数を計算することの困難さに依拠している。鍵選択には、g 及び x が p よりも小さくなるような、素数 p と2 個の乱数 g 及び x の選択が必要である。次に、 $y = g x \bmod p$ を計算する。公開鍵は、y、g 及び p である。秘密鍵は x である。

【1021】

暗号によるチャレンジャーレスポンスプロトコル及びゼロ知識証明

50

チャレンジレスポンスプロトコルの一般的な原理は、カメラシステムに適したアイデンティティ認証を提供することである。チャレンジレスポンスの最も簡単な形態は秘密パスワードの形態をとる。AはBに秘密パスワードを尋ね、Bが正しいパスワードで応答したとき、AはBが真正であることを表明する。この種の簡単なプロトコルには三つの主要な問題点がある。第一に、一旦Bがパスワードを公表すると、任意のオブサーバCはパスワードが何であるかを知る。第二に、Aはパスワードを検証するためにパスワードを知ることが必要である。第三に、CがAになりすました場合、Bは（CがAであると思い）Cにパスワードを与え、これにより、Bを危険にさらす。著作権のあるテキスト（例えば、俳句）を使用することは、弱い代用である。なぜならば、（例えば、知的所有権が遵守されない国では）誰でもパスワードをコピーすることができると考えられるからである。暗号によるチャレンジレスポンスプロトコルの考え方は、唯一のエンティティ（要求者）が、そのエンティティに関して知られている秘密情報の知識を実証することによって、プロトコル中に秘密情報自体を検証者に明かすことなく、自分のアイデンティティを他者（検証者）へ渡すことである。暗号によるチャレンジレスポンスプロトコルの一般化されたケースでは、ある種の方式により検証者はその秘密情報を知り、それ以外の場合には、その秘密情報は検証者にさえ知られていない。この実施形態の説明は、特に認証に関係しているので、認証の試し要される実際の暗号によるチャレンジレスポンスプロトコルは、適当なセクションで詳述される。しかし、ゼロ知識証明の概念はここで説明する。最初に、Feige、Fiat及びShamirによって明らかにされたゼロ知識証明プロトコルは、認証目的用のスマートカードで広く使用されている。このプロトコルの有効性は、未知の素因数分解による大きい合成整数の平方根モジュローの計算がコンピュータの能力で実施不可能である、という仮定に基づいている。これは、おそらく、大きい整数の素因数分解が困難であるという仮定と等価であろう。尚、要求者は著しい計算パワーをもっていなくてもよいことに注意する必要がある。スマートカードは、数回のモジュロー乗算だけを使用してこの種の認証を実現する。ゼロ知識証明プロトコルは米国特許第4747668号に記載されている。

【1022】

一方向関数

一方向関数Fは、入力Xに作用し、XがF[X]から判定できないようなF[X]を返す。Xのフォーマットに制限が無く、F[X]がXよりも少ないビットを格納する場合、衝突が存在する筈である。衝突は、同じF[X]値を生ずる2個の異なるX値、即ち、 $X_1 \neq X_2$ であり、しかも、 $F[X_1] = F[X_2]$ となるような X_1 及び X_2 として定義される。XがF[X]よりも多数のビットを格納する場合、入力は、出力を生成するために何らかの方法で圧縮されなければならない。殆どの場合に、Xは、特定のサイズのブロックに分割され、多数のラウンドに亘って圧縮され、あるラウンドの出力が次のラウンドへの入力である。このハッシュ関数の出力は、Xが使い尽くされた後の最後の出力である。圧縮関数CFの疑似衝突は、2個の異なる初期値 V_1 及び V_2 として定義され、2個の入力 X_1 及び X_2 （同一であるかもしれない）は、 $CF(V_1, X_1) = CF(V_2, X_2)$ となるように与えられる。疑似衝突が存在しても、所与の X_1 に対する X_2 の計算が簡単になるわけではないことに注意する必要がある。

【1023】

高速に計算できる一方向関数だけに関心がある。更に、様々な実装において再現可能である決定論的な一方向関数だけに関心がある。一例として、F[X]がFの呼び出しの間の時間であるFを考える。所与のF[X]に対して、Xは決定可能である。なぜならば、XはFによって均等に使用されないからである。しかし、Fからの出力は様々な実装において異なる。したがって、この種のFに興味はない。

【1024】

本実施形態の認証チップの実装の説明の範囲では、以下の形式の一方向関数：
未知鍵を使用する暗号化；
乱数シーケンス；

10

20

30

40

50

ハッシュ関数；
メッセージ認証コード；
だけに関心がある。

【1025】

未知鍵を使用する暗号化

メッセージが未知鍵 K を使用して暗号化されたとき、暗号化関数 E は事実上一方向性である。この鍵が無い場合、 $E_K[M]$ から M を K 無しに獲得することはコンピュータ能力的に実施不能である。暗号化関数は、鍵が隠されている限り一方向性である。暗号化アルゴリズムは衝突を生成しない。なぜならば、 E は、関数 D を使用して M を再構築できないような $E_K[M]$ を生成するからである。この結果として、 $F[X]$ は、一方向関数 F が E であるならば、少なくとも X と同数のビットを格納する（情報は失われない）。（上記の）対称暗号化アルゴリズムは、暗号化に基づく一方向関数を生成する非対称アルゴリズムよりも有利である。その理由は、
所定の強さの暗号化アルゴリズム用の鍵は、非対称アルゴリズムよりも対称アルゴリズムの方が短くなり、
対称アルゴリズムの方が高速に計算でき、必要なソフトウェア／シリコンが少ない、からである。

10

【1026】

優れた鍵の選択は選択された暗号化アルゴリズムに依存する。ある種の鍵は特定の暗号化アルゴリズムの場合に強くないので、全ての鍵の強さをテストすることが必要である。鍵選択のために実行されるべきテストの回数が増えるほど、その鍵が隠された状態を保つ可能性が低くなる。

20

【1027】

乱数シーケンス

乱数シーケンス $R_0, R_1, \dots, R_i, R_{i+1}$ を考える。 $F[X]$ が乱数シーケンス内の X 番目の乱数を返すような一方向関数を定義する。しかし、 $F[X]$ は様々な実装において所定の X に関して再現可能であることを保証しなければならない。したがって、乱数シーケンスは、本当の意味でランダムではない。その代わりに、乱数シーケンスは疑似乱数であり、生成器は特殊なシードを利用する。

【1028】

優れた乱数生成器の定義に関して多数の問題がある。 $Knuth$ は、生成器を「良くする」要因（統計的テストを含む）、生成器の構築に関する一般的な問題を説明している。大半の乱数生成器は、 $i-1$ 番目の状態から i 番目の乱数を生成し、 i 番目の数を決定するための唯一の方法は、 0 番目の数から i 番目の数までを繰り返すことである。 i が大きい場合、 i 回の繰り返しを待つことは実際的ではない。しかし、あるタイプの乱数生成器はランダムアクセスを許可しない。 $Blum$ 、 $Blum$ 及び $Shub$ は、理想的な生成器に関して、「短いシードから、校正なコインの連続的な反転によって完全に発生させられたように見える長い（ビットの）シーケンスを素早く生成するためには疑似ランダムシーケンス生成器が好ましい」のように定義している。彼らは、 X^2

30

$\text{mod } n$ 型の生成器、一般的には、 BBS 生成器と呼ばれる生成器を定義した。現在の暗号化方式が依拠しているある種の仮定を前提とした場合、 BBS 生成器は非常に厳しい統計的テストに合格することを示した。

40

【1029】

BBS 生成器は、 $Blum$ 整数である n の選択に依拠している（ $n = pq$ であり、 p 及び q は大きい素数であり、 $p \neq q$ であり、 p

$\text{mod } 4 = 3$ であり、 q

$\text{mod } 4 = 3$ である）。生成器の初期状態は、 x_0 によって与えられ、ここで、 $x_0 = x^2$

$\text{mod } n$ であり、 x は、 n よりも相対的に優良なランダム整数である。 i 番目の疑似ランダムビットは、 x_i の最下位ビットであり、 $x_i = x_{i-1}^2$

50

$\text{mod } n$ である。付加的な性質として、 p 及び q の知識は、 $x_i = x_0^y$
 $\text{mod } n$ 、但し、 $y = 2^i$

$\text{mod } ((p-1)(q-1))$ によって、シーケンス内の i 番目の数の直接的な計算
 を可能にさせる。

【1030】

p 及び q の知識がない場合、生成器は、繰り返しを実行しなければならない（計算の安全性は大きい数の素因数分解の困難さに依存する）。当初定義されたとき、BBS生成器に関する主要な問題は、単一の出力ビットのために必要な作業量であった。このアルゴリズムは、殆どのアプリケーションに対して遅すぎると考えられていた。しかし、モンゴメリ型リダクション演算の出現によって、より実際的な実装形態が生じた。その上、Vaziri 10
 rani 及び Vazirani は、 n のサイズに依存して、生成器の安全性を譲歩することなく、より多数のビットを x_i から安全に取り出せることを示した。 x_i 毎に1ビット
 だけが必要であるとするならば、 N ビット（したがって、ビット生成器関数の N 回の反復）が N ビット乱数を発生させるために必要である。外部オブザーバーにとって、特定のビ
 ットの組が与えられた場合に、半々の確率以外で次のビットを決定する方法は無い。 x 、
 p 及び q が秘密にされている場合、それらは鍵として作用し、出力ビットストリームを取り出し、 x 、 p 及び q を計算することはコンピュータ能力的に実施不能である。また、所
 定の疑似ランダムビットの組を発生させるために使用された i の値を決定することは、コ
 ンピュータ能力的に実施不能である。この最後の特徴によって、生成器は一方方向性になる。異なる i の値が、所定の長さの同じビットシーケンス（例えば、32 ビットのランダム 20
 ビット）を生じさせる可能性がある。たとえ、 x 、 p 及び q が既知であるとしても、予定の $F[i]$ に対して、 i は確率の組として導出できるだけであり、特定の値として導出で
 きない。（勿論、 i のドメインが既知であるならば、確率の組は更に限定される）。しかし、良好な p 及び q と、良好なシード x の選択に関して問題がある。特に、Ritter
 は、 x を選択する際の問題について記述している。この問題の性質は、BBS生成器が既
 知長さの単一のサイクルを生成しないことである。その代わりに、BBS生成器は、縮退
 （長さゼロ）サイクルを含む様々な長さのサイクルを生成する。このように、BBS生成
 器は、短いサイクルかもしれないランダム状態を用いて初期化することが不可能である。

【1031】

ハッシュ関数

ハッシュ関数として知られている特殊な一方方向関数は、任意長さのメッセージを固定長さ
 のハッシュ値へマップする。ハッシュ関数は $H[M]$ のように表される。入力任意長さ
 であるため、ハッシュ関数は固定長さの出力を生ずるため圧縮コンポーネントを有する。
 また、ハッシュ関数は、衝突の検出を困難にさせ、 $H[M]$ から M に関する情報の決定を
 困難にさせるため、オブファスケーションコンポーネントを含む。衝突は実際に存在する
 ので、殆どのアプリケーションは、所与の X_1 に対して、 $H[X_1] = H[X_2]$ となる
 ような X_2 を見つけることが困難であるという点で、ハッシュアルゴリズムが耐プレイメ
 ージ性であることを必要とする。その上、殆どのアプリケーションは、ハッシュアルゴリ
 ズムが耐衝突性である（即ち、 $H[X_1] = H[X_2]$ となるような2個のメッセージ X_1 及び X_2 を見つけることが困難である）ことを要求する。耐衝突性のあるハッシュ関数 40
 が理想的な意味で存在し得るかどうかは、未確定の問題である。ハッシュ関数の主なアプ
 リケーションは、デジタル署名アルゴリズムを適用する前に、入力メッセージをデジタル
 「指紋」に変形することである。デジタル署名との衝突の一つの問題は以下の例に示され
 ている。

【1032】

A は、「私は B に 100 ドルの借りがある。」という長いメッセージ M_1 を持っている。
 A は、自分の秘密鍵を用いて $H[M_1]$ を署名する。どんな B は、 $H[M_2] = H[M_1]$
 であり、 M_2 が B にとって有利であるような衝突メッセージ M_2 、例えば、「私は B
 に 100 万ドルの借りがある。」を探索する。明らかに、このような M_2 を見つけること
 が困難であることを保証することは A の利益である。

10

20

30

40

50

【1033】

耐衝突性のある一方向ハッシュ関数の例は、SHA-1、MD5、及びRIPEMD-160であり、これらは全てMD4から導かれる。

【1034】

MD4

Ron Rivestは1990年にMD4を発表した。他の全ての一方向ハッシュ関数はMD4から何らかの方法によって導出されるので、ここでは、MD4を説明する。MD4は、衝突を探索されるのではなく、計算可能であるという点で、現在では、完全に破られたと考えられている。上記の例において、Bは、元のメッセージ M_1 と同じハッシュ値を用いて、代わりのメッセージ M_2 を普通に発生させることができる。

10

【1035】

MD5

Ron Rivestは1991年に、安全性の高まったMD4として、MD5を発表した。MD4と同様に、MD5は128ビットのハッシュ値を生成する。Dobbertinは、最近の攻撃後のMD5の状態を説明している。彼は、MD5において疑似衝突を見つける方法を説明し、圧縮関数の弱点を指摘し、つい最近、衝突が発見された。即ち、MD5は、衝突の存在がひどい結果を招くデジタル署名スキームの圧縮に使用されるべきではない。しかし、MD5は、一方向関数として依然として使用することができる。その上、HMAC-MD5の構造は、これらの最近の攻撃による影響を受けない。

20

【1036】

SHA-1

SHA-1はMD5と非常に類似しているが、160ビットのハッシュ値を有する（MD5は128ビットのハッシュ値しかない）。SHA-1は、デジタル署名標準（DSS）で使用するため、NIST及びNSAによって設計され発表された。最初に発表された説明はSHAと呼ばれたが、その後直ぐに、おそらくSHAの安全性の欠陥を訂正するために改訂され、SHA-1となった（但し、NSAは変更の原因となった数学的な理由を発表していない）。SHA-1に対する暗号攻撃は知られていない。また、SHA-1は、単にハッシュ結果が長くなったという理由で、力づくの攻撃に対する耐性がMD4又はMD5よりも強い。米国政府は、SHA-1及びDSAアルゴリズム（DSSの一部として定められたデジタル署名認証アルゴリズム）を保有し、少なくとも一つの関連特許（1993年に発行された米国特許5231688号）を保持する。

30

【1037】

RIPEMD-160

RIPEMD-160は、その前進であるRIPEMD（1992年に欧州委員会のRIPEプロジェクトのために開発された）から導出されたハッシュ関数である。その名前から分かるように、RIPEMD-160は、160ビットのハッシュ結果を生成する。32ビットアーキテクチャのソフトウェア実装のため合わされたRIPEMD-160は、10年間以上に亘って高いセキュリティレベルを提供することを予定されている。RIPEMD-160に対する攻撃の成功例は存在しないが、RIPEMD-160は比較的新しく、大規模に解読されていない。最初のRIPEMDアルゴリズムは、従来のMD4に対する暗号攻撃に対抗するために特に設計された。最近のMD5に対する攻撃は、RIPEMDの128ビットハッシュ関数においても類似した弱点を明らかにした。この攻撃は理論的な弱点を示しただけであるが、Dobbertin、Preneel及びBosselaersは、RIPEMDを新しいアルゴリズムRIPEMD-160へ更に強化した。

40

【1038】

メッセージ認証コード

メッセージ認証の問題は以下のようにまとめることができる：

Aは、Bからのメッセージであると推測されるメッセージが本当にBからのメッセージであることをどのようにして確信するか？

50

メッセージ認証はエンティティ認証とは異なる。エンティティ認証の場合、一つのエンティティ（要求者）は、自分のアイデンティティを他者（検証者）に対して照明する。メッセージ認証の場合、所定のメッセージが自分の思う人からのメッセージであることを保証すること、即ち、メッセージが発信元から宛先までの途中で改ざんされていないことを保証することに関係する。一方向ハッシュ関数はメッセージに対する保護が十分ではない。MD5のようなハッシュ関数は、元の入力を代表するハッシュ値を発生させ、元の入力がハッシュ値から導出できないことに基づいている。AとBの間にいるEからの簡単な攻撃は、Bからのメッセージを途中で盗み、自分のメッセージで置き換えることである。たとえば、Aが元のメッセージのハッシュを送信したとしても、Eは自分の新しいメッセージのハッシュで簡単に置き換えることができる。一方向ハッシュだけを使用した場合、AはBのメッセージが変更されたかどうかを知る方法がない。メッセージ認証の問題に対する一つのソリューションは、メッセージ認証コード、即ち、MACだえる。BがメッセージMを送信するとき、Bは、受信者にMが本当にBからのメッセージであることがわかるようにMAC [M]を送信する。これを実現するため、BだけがMのMACを生成可能であり、更に、AはMAC [M]と対照してMを検証し得ることが必要である。これは、Mを秘密にする必要がないときに、M-MACの暗号化が有効であることとは異なることに注意する必要がある。ハッシュ関数からMACを構築する最も簡単な方法は、対称アルゴリズムを用いてハッシュ値を暗号化すること、即ち、

入力メッセージをハッシュ化 $H[M]$

ハッシュを暗号化 $E_K[H[M]]$

である。

【1039】

これは、最初にメッセージを暗号化して、次に、暗号化されたメッセージをハッシュ化するよりも安全性が高い。任意の対称又は非対称暗号化関数を使用することができる。しかし、（上記のような）暗号化を使用する技術の代わりに、鍵依存型の一方向ハッシュ関数を使用する方が以下の幾つかの点で有利である：

速度：一方向ハッシュ関数は一般的に暗号化よりも高速に動作するため；

メッセージサイズ： $E_K[H[M]]$ は少なくともMと同じサイズであるが、 $H[M]$ は固定サイズ（通常、Mよりもかなり小さい）であるため；

ハードウェア／ソフトウェア必要条件：鍵付きの一方向ハッシュ関数は、典型的に、それらの暗号ベースの代案よりも遙かに複雑さが抑えられている；

一方向ハッシュ関数の実装は、暗号化装置又は復号化装置であるとはみなされないので、米国の国外持ち出し規制の対象ではない。

【1040】

尚、ハッシュ関数は、当初は、鍵を格納したり、メッセージ認証をサポートしたりするためには設計されていなかったことに注意する必要がある。その結果として、メッセージを秘密プレフィックス、サフィックス、又は両方と連結する種々の関数を含む、メッセージ認証を実行するためにハッシュ関数を使用する、ある種のアドホックな方法が提案されている。これらのアドホックな方法の殆どは、洗練された手段によって巧く攻撃された。付加的なMACは、XORスキーム及びテブリッツ（Toeplitz）行列に基づいて提案されている（LFSRベース構造の特殊なケースを含む）。

【1041】

HMAC

特にHMAC構造は、インターネットメッセージ認証セキュリティプロトコル用のソリューションとして認められ始めている。HAMC構造は、基礎となるハッシュ関数をブラックボックス的に使用して、ラッパーとして作用する。ハッシュ関数の置換は、安全性又は性能上の理由から求められるならば、簡単に行われる。しかし、HMAC構造の主要な利点は、基礎となるハッシュ関数がある程度の合理的な暗号強度を有するならば、HMAC構造は安全であることが証明できること、即ち、HMACの強度は、ハッシュ関数の強度と直接的に関連していることである。HMAC構造はラッパーであるため、任意の反復ハ

10

20

30

40

50

ッシュ関数をHMACで使うことが可能である。その例には、HMAC-MD5、HMAC-SHA1、HMAC-RIPMD160等が含まれる。以下のような定義：

H=ハッシュ関数（例えば、MD5又はSHA-1）

n=Hから出力されたビット数（例えば、SHA-1の場合160ビット、MD5の場合128ビット）

M=MAC関数が適用されるべきデータ

K=2当事者によって共有される秘密鍵

ipad=0x36の反復64回

opad=0x5Cの反復64回

を与えると、HMACアルゴリズムは：

0x00バイトをKの最後に付加することによってKを64バイトへ拡張する；

（1）で作成された64バイト文字列をipadでXOR演算する；

（2）で作成された64バイトにデータストリームMを付加する；

（3）で発生されたストリームにHを適用する；

（1）で作成された64バイト文字列をopadでXOR演算する；

（4）からの結果Hを（5）からの結果である64バイトに付加する；

（6）の出力にHを適用し、結果を出力する；

のようになる。したがって、

$HMAC[M] = H[(KAopad) | H[(KAipad) | M]]$

である。

【1042】

推奨される鍵長さは少なくともnビットであるが、64バイト（ハッシュ化ブロックの長さ）よりも長くなってはならない。nビットよりも長い鍵は、関数の安全性を上乗せしない。HMACは、オプションとして、最終的な出力の丸め、例えば、160ビットから128ビットへの丸めを行うことできる。HMAC設計者の規約（RFC）は、このアルゴリズムが最初に発表された1年後の1997年に発行された。設計者は、HMACに対する既知の最強の攻撃は、ハッシュ関数Hに対する衝突の頻度に基づく攻撃であり、最低限度で適当なハッシュ関数に対して全く实际的ではないということを公表している。より最近では、リプレイ防止機能付きHMACプロトコルが、所定の時間内でのM、HMAC[M]コンビネーションの捕捉及び再生を防止するために規定されている。

【1043】

乱数及び時変性メッセージ

一方向関数としての乱数生成器の使用については既に説明した。しかし、乱数生成器の理論は、暗号法、セキュリティ、及び認証と非常に絡み合っている。優れた乱数生成器の定義に関して多数の問題がある。Knuthは、生成器をよくするもの（統計的テストを含む）、及び生成器の構築に関する一般的な問題を説明している。乱数の用法の一つは、メッセージが時間的に変化することを保証することである。Aがコマンドを暗号化し、それをBへ送信するシステムを想定する。暗号化アルゴリズムが所与の入力と同じ出力を生成する場合、攻撃者は、簡単にメッセージを記録し、Bを欺くためにそれを再生することが可能である。攻撃者は、暗号化機構を破壊する必要がなく、（Aになりすましている間に）Bに対して再生すべきメッセージがわかればよい。その結果として、メッセージは、しばしば、そのメッセージ（したがって、その暗号化された対応部分）が常に変化することを保証するため、乱数及びタイムスタンプを含む。乱数生成器は、しばしば、鍵を生成するためにも使用される。したがって、現時点では、全ての生成器は、この目的のためには不確かであると言って差し支えない。例えば、Berlekamp-Masseyアルゴリズムは、LFSR乱数生成器に対する従来の攻撃法である。LFSRの長さがnであるならば、2nビットのシーケンスは、鍵生成器を構成するLFSRを決定するために十分である。しかし、乱数生成器の唯一の役目は、メッセージが時間的に変化することを保証することであるならば、生成器及びシードのセキュリティは、セッション鍵生成の場合のように重要ではない。しかし、乱数シード生成器が危険にさらされ、攻撃者が未来の「

乱数」を計算可能であるならば、一部のプロトコルが攻撃者に対して開放されたままにされ得る。新しいプロトコルはこの状況に関して検討されるべきである。要求される実際のタイプの乱数生成器は、インプリメンテーションと、その生成器の使用目的と、に依存するであろう。生成器には、R o n

R i v e s t による R C 4 のような B l u m、B l u m、及び S h u b ストリーム暗号、S H A - 1 及び R I P E M D - 1 6 0 のようなハッシュ関数、並びに、L F S R（線形フィードバックシフトレジスタ）及びそれらのより最近の対応物である F C S R（キャリア付きフィードバックシフトレジスタ）のような従来の生成器が含まれる。

【1044】

攻撃

このセクションは、認証チップのような認証暗号システムを破るために行われる可能性のある様々なタイプの攻撃を説明する。攻撃は、物理的な攻撃と論理的な攻撃に大別される。物理的な攻撃は、暗号システムの物理的な実装を破壊する（例えば、鍵を取り出すためにチップを壊して開ける）方法を表し、論理的な攻撃は、実装に依存しない暗号システムへの攻撃を含む。論理的なタイプの攻撃は、プロトコル又はアルゴリズムに対して機能し、以下の三つのうちの何れかを実行しようとする：

認証プロセスを完全に回避する；

あらゆる質問に回答できるように、力づく又は推論によって秘密鍵を取得する；

鍵を用いることなく各質問に正しい回答を与えるため、認証用の質問及び回答の性質を十分に検出する。

【1045】

次に、攻撃のスタイル及び形式を詳細に説明する。セキュリティチップによって使用されるアルゴリズム及びプロトコルとは無関係に、チップの認証部の回路は物理的な攻撃を受ける可能性がある。物理的な攻撃は、以下の4つの主要な形で現れるが、攻撃の形態は変化し得る：

認証チップを完全に回避する；

動作中にチップを物理的に検査する（破壊的及び非破壊的）；

チップを物理的に分解する；

チップを物理的に改変する。

【1046】

次に、攻撃のスタイル及び形式を詳細に説明する。このセクションは、これらの攻撃に対するソリューションを提案するものではない。このセクションは、各攻撃タイプを説明するだけである。調査は、（インターネット認証のような他のある種のシステムではなく）あるシステムに取り付けられた認証チップ53の状況に限定されている。

【1047】

論理的な攻撃

これらの攻撃は、暗号システムの物理的な実装形態に依存しない。これらは、プロトコル、アルゴリズムの安全性、及び乱数生成器に作用する。

【1048】

暗号文だけの攻撃

これは、攻撃者が、全て同じアルゴリズムを使用して暗号化された一つ以上の暗号化メッセージをもつ場合である。攻撃者の目的は暗号化メッセージから平文メッセージを獲得することである。理想的には、鍵を復元することが可能であり、その結果、将来のすべてのメッセージを復元可能である。

【1049】

既知平文攻撃

これは、攻撃者が、平文と、平文の暗号化形式の両方をもつ場合である。認証チップの場合、既知の平文の攻撃は、攻撃者がシステムと認証チップの間のでたフローを見ることが出来る場合である。入力及び出力は（攻撃者によって選別されることなく）観測され、弱点を見つけるため解析され得る（例えば、バースデー攻撃、又は区別の付く興味ある入

10

20

30

40

50

力／出力のペアの探索によって）。既知平文攻撃は、選択平文攻撃よりも弱いタイプの攻撃である。なぜならば、攻撃者はデータフローしか観測できないからである。既知平文攻撃は、ロジックアナライザを、システムと認証チップの間の配線に接続することによって実行し得る。

【1050】

選択平文攻撃

選択平文攻撃は、暗号解読者が任意の選択されたメッセージを暗号システムへ送信し、その応答を観測できる能力を備えている場合である。暗号解読者がアルゴリズムを知っているならば、特定の出力を別の関数の入力へ供給することによって利用可能な入力と出力の間の関係があるであろう。内蔵型認証チップを使用するシステムでは、選択平文攻撃を防止することは難しい。なぜならば、暗号解読者は、論理的に自分がシステムであるふりをして、任意の選択されたビットパターンストリームを認証チップへ送信できるからである。

10

【1051】

適応的選択平文攻撃

このタイプの攻撃は選択平文攻撃と類似しているが、攻撃者が先行の試験の結果に基づいて後続の選択された平文テキストを変更し得る点で異なる。上述のシステム／認証チップのシナリオがコピー機及びトナーカートリッジのような消耗品のために利用されるときは、確実にこのケースである。なぜならば、特に、システムと消耗品の両方は誰でも入手できるようにされているからである。

20

【1052】

ブルートフォース攻撃

鍵ベースの暗号システムアルゴリズムを破るための保証付きの方法は、単純に全ての鍵を試してみることである。最終的に正しい鍵が見つけれらるであろう。これは、ブルートフォース攻撃として知られている。しかし、鍵候補の数が増加すると、より多くの鍵を試してみる必要があり、正しい鍵を見つけるために（平均的に）要する時間が長くなる。N個の鍵がある場合、最大でN回の試行が必要である。鍵の長さがNビットであるならば、最大で 2^N 回の試行が必要であり、半分だけの試行（ 2^{N-1} ）後に鍵を見つけるチャンスは50%である。Nが長くなればなるほど、鍵を見つけるために要する時間が長くなるので、鍵の安全性が高まる。勿論、攻撃が最初の試行で鍵を当てる可能性はあるが、この可能性は鍵が長くなると共に低下する。ここで、56ビットの鍵長さを考える。最悪ケースでは、鍵を見つけるために全部で 2^{56} 回のテスト（ 7.2×10^{16} 回のテスト）を行わなければならない。1977年に、DiffieとHellmanは、100万個のプロセッサにより構成され、各プロセッサが每秒100万回のテストを実行刷る能力を備えたDES解読専用機械を発表した。このような機械は任意のDESコードを破るためには20時間を要する。鍵長さが128ビットの場合を考える。最悪ケースでは、鍵を見つけるために全部で 2^{128} 回のテスト（ 3.4×10^{38} 回のテスト）を行わなければならない。これは、各々が每秒10億回のテストを実行する1兆個のプロセッサ上で10億年を要する。鍵長さが十分に長い場合、ブルースフォース攻撃は非常に長い時間を要するので、攻撃者の努力に値しない。

30

40

【1053】

推測攻撃

このタイプの攻撃は、攻撃者が単に鍵を「推測」しようとする場合である。攻撃としては、ブルースフォース攻撃と同じであり、成功の確率は鍵の長さにかかっている。

【1054】

量子コンピュータ攻撃

nビット鍵を解読するため、適切なアルゴリズムに埋め込まれたnキュビットを有する量子コンピュータ（NMR、オプティカル、又はケージドアトム）を構築しなければならない。量子コンピュータは、実効的に、 2^n 個の同時コヒーレント状態に存在する。デコヒーレントを生じることなく正しいコヒーレント状態を抽出することが要領である。今まで

50

のところ、これは、2 キュビットのシステム（4 コヒーレント状態で存在する）によって達成されている。数年の間に、これを6 キュビット（64 の同時コヒーレント状態をもつ）まで拡張することは可能であると思われる。

【1055】

残念なことに、キュビットを付加する毎に、鍵を表現する信号の相対強度が半分になる。これは、特に、暗号方式的に安全なシステムで使用される長い鍵の場合に、直ぐに鍵取り出しの重大な障害になる。その結果として、量子コンピュータを使用した、暗号方式的に安全な鍵（例えば、160 ビット）に対する攻撃は、おそらく実施される可能性はなく、量子コンピュータが、認証チップの商業的な耐用年限中に50 キュビット以上のキュビット数を達成する可能性は非常に低い。たとえ、50 キュビットの量子コンピュータを使用しても、160 ビット鍵を解読するために、 2^{160} 回のテストが必要である。

10

【1056】

意図的なエラー攻撃

一部のアルゴリズムの場合、攻撃者は悪い入力の結果から有用な情報を収集することができる。有用な情報の範囲は、エラーメッセージ文から、エラーを発生させるために要した時間までに亘る。簡単な一例は、ユーザID／パスワードのスキームである。エラーメッセージが通常「間違ったユーザID」であるならば、攻撃者が「間違ったパスワード」というメッセージを取得したとき、ユーザIDが正しいことを知る。メッセージが常に「間違ったユーザID／パスワード」であるならば、攻撃者に与えられる情報は遙かに少ない。より複雑な例は、最近発表された、安全なウェブサイトから暗号コードを解読する方法である。この攻撃には、特定のメッセージをサーバーへ送信することと、エラーメッセージ応答を観測することとが含まれる。応答は、鍵を学習するために十分な情報を与え、応答が無くても何らかの情報が得られる。アルゴリズムの一例として、時間は、誤りのあるビットが入力メッセージ中に検出されたときに直ちにエラーを返すアルゴリズムの場合に知ることができる。ハードウェア実装形態に依存して、攻撃者が、応答の時間を計り、エラー応答に要した時間に基づいて1 ビットずつ変更し、このようにして鍵を獲得することは、簡単な方法である。チップ実装形態の場合、必要な時間がインターネット経由の場合よりもかなり高い精度で観測できることは確実である。

20

【1057】

バースデー攻撃

この攻撃は、有名な「バースデーパラドックス」（実際には全くパラドックスではない）にちなんで命名された。ある人が他の人と同じ誕生日である確率は365分の1である（閏年は考えない）。したがって、室内の183人のうちの一人があなたと同じ誕生日である確率は50%よりも高い筈である。しかし、室内に23人の人がいれば、何れか二人の誕生日が同じである確率は50%を超える。その理由は、23人によって253通りのペアが生じるからである。バースデー攻撃は、ハッシングアルゴリズム、特に、ハッシングとデジタル署名を組み合わせたアルゴリズムに対する一般的な攻撃である。メッセージが生成され、既に署名されている場合、攻撃者は、同じ値にハッシュする衝突メッセージを探す必要がある（あなたと同じ誕生日の人を見つけることと類似している）。しかし、攻撃者がメッセージを生成可能であるならば、バースデー攻撃が作用し始める。攻撃者は、同じハッシュ値をもつ二つのメッセージを探し（何れか二人が同じ誕生日である場合と類似している）、一方のメッセージだけがそれに署名をした人に受け入れられ、もう一方は攻撃者の役に立つ。その人が元のメッセージに署名をし終えたならば、攻撃者は、その人は別のメッセージに署名をしたことを主張するだけでよく、数学的にどちらのメッセージが原本であるかを示す方法はない。なぜならば、両方のメッセージは同じ値にハッシュするからである。ブルートフォース攻撃が一致を判定するための唯一の方法であるとする、バースデー攻撃によるnビット鍵の脆弱化は $2^{n/2}$ である。バースデー攻撃を受ける可能性のある128ビットの長さの鍵は、事実上64ビットの長さしかない。

40

【1058】

連鎖攻撃

50

これらの攻撃は、ハッシュ関数の連鎖性に対して行われる攻撃である。連鎖攻撃は、ハッシュ関数の圧縮関数に的を絞る。このアイデアは、ハッシュ関数が一般的に任意の長さの入力を取り、入力 n ビットを同時に処理することによって、一定の長さの出力を生成する点に基づいている。1 ブロックからの出力は次のブロックへの連鎖変数セットとして使用される。入力全体に対する衝突を見つけるのではなく、このアイデアは、入力連鎖変数セットが与えられた場合に、本来のメッセージと同じ出力連鎖変数を生じる代用ブロックを見つけることである。特定のブロックの選択数は、ブロックの長さに基づいている。連鎖変数が c ビットであり、ハッシング関数はランダムマッピングのように挙動し、ブロック長さが b ビットであるならば、このような b ビットブロックの個数は、約 $2^b / 2^c$ 個である。代用ブロックを見つけるためのチャレンジは、このようなブロックが考えられる全てのブロックのうちの疎部分集合であることである。SHA-1 の場合、512 ビットブロックの個数は、約 $2^{512} / 2^{160}$ 、即ち、 2^{352} 個である。ブルートフォース検索によってブロックを見つける見込みは 2^{160} 分の 1 である。

10

【1059】

完全ルックアップテーブルによる代用

チップへ送信された可能性のあるメッセージの数が少ない場合、鍵を解読するために複製品製造者は不要である。その代わりに、複製品製造者は、システムによって送信されたコードに対する純正チップからのレスポンスの全てを記録した ROM をチップに組み込むことが可能である。鍵が長くなり、レスポンスが長くなると共に、このようなルックアップテーブルに必要な空間が大きくなる。

20

【1060】

疎ルックアップテーブルによる代用

チップへ送られたメッセージが、実効的にランダムではなく、ある程度予測可能であるならば、複製品製造者は完全ルックアップテーブルを提供しなくてもよい。例えば：

メッセージが単なるシリアル番号であるならば、複製品製造者は、過去及び未来の予測シリアル番号を格納するルックアップテーブルを提供すればよい。これらのシリアル番号が 10^{19} 個を超える可能性は低い；

テストコードが単なる日付であるならば、複製品製造者はアドレスとして日付を使用してルックアップテーブルを生成することができる；

テストコードが、シリアル番号又は日付をシードとして用いる疑似乱数であるならば、複製品製造者は、まさに、システム内の疑似乱数生成器を解読する必要がある。これは、おそらく難しくはないであろう。なぜならば、複製品製造者はシステムのオブジェクトコードを手に入れるからである。複製品製造者は、蓄積された認証コードへアクセスするためこれらのコードを使用して、内容アドレス可能メモリ（又は、その他の疎配列ルックアップ）を生成するであろう。

30

【1061】

差分暗号解読

差分暗号解読は、入力ストリームのペアが既知差分を用いて生成され、符号化ストリームの差分が解析される攻撃を表す。既知差分攻撃は、DES 及びその他の同様のアルゴリズムで使用されるような S ボックスの構造にかなり依存している。HMAC-SHA1 のような他のアルゴリズムは S ボックスを持たないが、攻撃者は、

40

最小差分入力及びそれに対応した出力

最小差分出力及びそれに対応した入力

の統計的解析を行うことによって差分的な攻撃を実行し得る。

【1062】

殆どのアルゴリズムは、差分暗号解読のプロセスが記載された後、差分暗号解読に対して補強された。これは、各暗号化方式に専用のセクションで対称とされている。しかし、一部の密かに開発された近年のアルゴリズムは破られている。なぜならば、開発者がある種のスタイルの差分攻撃を考慮せず、自分のアルゴリズムを公開の精査の対象としなかったからである。

50

【1063】

メッセージ置換攻撃

ある種のプロトコルでは、媒介物はメッセージの一部又は全部を置換可能である。これは、真の認証チップが消耗品内の再使用可能な複製チップにプラグインされる場合である。複製チップは、システムと認証チップの間の全てのメッセージを盗み取り、多数の置換攻撃を実行し得る。ヘッダとその後続く内容とを格納したメッセージを例として考える。攻撃者は有効ヘッダを生成し得ないが、特に、有効レスポンスが「はい、あなたのメッセージを受信しました。」のようなものであるならば、メッセージの固有の内容を置換することができる。たとえ、リターンメッセージが「はい、以下のメッセージを受信しました...」であったとしても、攻撃者は、肯定応答を本来の送信者へ返信する前に、元のメ

10

【1064】

鍵生成器のリバースエンジニアリング

疑似乱数生成器が鍵を生成するため使用される場合、複製品製造者は、生成器プログラムを取得するか、又は使用されたランダムシードを推測する可能性がある。これは、ネットスケープのセキュリティプログラムが最初に破られたときの方法である。

【1065】

認証全体の回避

認証プロトコルには認証プロセス全体の回避を可能にする問題があり得る。この種の攻撃の場合、鍵は全く無関係であり、攻撃者は鍵を復元する、又は鍵を推定する必要がない。電源投入時に認証を行なうが、その他のときには認証しないシステムを例として考える。複製認証チップを含む再使用可能な消耗品は、真正認証チップを利用する場合もある。複製認証チップ53は、認証呼び出しのため真正チップを使用し、次に、その後の真正認証チップの状態データをシミュレーションする。認証を回避する別の例は、消耗品が使用された後に限りシステムが認証する場合である。複製認証チップは、消耗品の使用後で、認証プロトコルが完了する前に（或いは、開始される前に）コネクションの失敗をシミュレーションすることにより単純な認証回避を実行できる。「ケンタッキーフライドチップ」ハックと呼ばれる悪名の高い攻撃は、衛星テレビジョンシステム用のマイクロコントローラチップを置換する。加入者が受信料の支払いを停止したとき、システムは、「使用不能」メッセージを送出するであろう。しかし、新しいマイクロコントローラは、このメッセージを検出するだけで、それを消費者の衛星テレビジョンシステムへ渡さない。

20

30

【1066】

強盗／賄賂攻撃

鍵を知っている人は、それを他の誰かに教える可能性がある。この秘密の暴露は、強制（賄賂、強盗等）、報復（例えば、不満を抱いている従業員）、又は単に信条のために行われる。これらの攻撃は、通常、鍵を推測する他の試みよりも安価かつ容易である。一例として、D i v x規格の開発に関わることを要求した多数の人々は、近年（1998年5／6月）、D i v x仕様解説解説装置の開発に参加したいという趣旨で、即ち、信条によって、多種多様なDVDニュースグループで主張している。

40

【1067】

物理的攻撃

以下の攻撃は、攻撃者が物理的にアクセスできるシリコンチップに認証機構が実装されていることを前提としている。最初の攻撃であるROM読み出しは、鍵がROMに保存されているときの攻撃を表し、残りの攻撃は秘密鍵がフラッシュメモリに保存されることを前提としている。

【1068】

ROM読み出し

鍵がROMに保存されている場合、鍵を直接読み出すことが可能である。ROMは、このように（非対称暗号方式で使用するための）公開鍵を保持するため安全に使用することが

50

できるが、秘密鍵を保持するためには使用できない。対称暗号方式では、ROMは全く当てにならない。著作権付きテキスト（例えば、俳句）を鍵として使用することは不十分である。なぜならば、鍵の複製は、知的所有権が遵守されない国で行われることが想定されるからである。

【1069】

チップのリバースエンジニアリング

チップのリバースエンジニアリングでは、攻撃者は、チップを分解し、回路を解析する。回路が解析されてしまうと、チップのアルゴリズムの内部動作を復元することができる。ルーセント・テクノロジー社は、回路を映像化するため、TOBIC（ツー・フォトンOBICの略であり、OBICはOptical Beam Induced Current光ビーム誘導電流の頭文字である。）と呼ばれるアクティブな方法を開発した。主としてスタティックRAM解析のために開発されたプロセスは、全ての裏材を取り除き、裏面を鏡面仕上げまで磨き、表面に光を集光させる。励起波長は、特にICに電流を誘導しないように選択される。19世紀にケルクホフスは、アルゴリズムの内部動作がスキームの単なる秘密であるならば、スキームは解読されたも同然である、という暗号解読に関する基本的な仮説を立てた。ケルクホフスは、秘密性は完全に鍵の中に存在しなければならない、ということを明記している。その結果として、チップのリバースエンジニアリングに対する最良の保護策は、内部動作を無関係にすることである。

10

【1070】

認証プロセスの不当使用

あらゆる複製品製造者はシステムと消耗品の両方のデザインを入手していることを想定しなければならない。同じチャンネルが、システムと信頼できるシステム認証チップとの間、並びに、信頼できない消耗品認証チップとの間の通信のために使用されるならば、信頼できないチップが「正しい回答」を得るために信頼できるチップに問い合わせをする可能性がある。もしそうであるならば、複製品製造者は鍵を決定する必要があるであろう。彼らは、システム認証チップからの応答を使用してシステムをだますだけでよい。認証プロセスを不当使用する別の方法は、論理的攻撃「認証プロセスの回避」と同じ方法に従って、認証プロセスが行われるとき、システムとの接続の失敗をシミュレーションし、電源遮断等をシミュレーションする。

20

30

【1071】

システムの変更

この種の攻撃は、システム自体が複製消耗品を受け入れるように変更される攻撃である。攻撃は、システムROMの変更、消耗品の配線し直し、又は極端なケースでは、完全な複製システムである。この種の攻撃は、各個のシステムが変更されることが必要であり、殆どの場合に所有者の承諾が必要であろう。通常は、消費者がこのような変更を行うために明らかな利益が必要である。なぜならば、このような変更は、典型的に、保証を無効にさせ、殆どの場合に費用が掛かるからである。消費者にとって明らかな利益のあるこのような変更の一例は、固定リージョン型DVDプレーヤーをリージョンフリー型DVDプレーヤーに変更するソフトウェアパッチである。

40

【1072】

従来のプロベリングによるチップ動作の直接視

チップ動作がSTM又は電子ビームを使用して直接監視できるならば、鍵は、内部不揮発性メモリから読み出され、作業レジスタへ取り込まれるときに記録できる。従来 of 3形態のプロベリングは、ICの通電中にICの上面又は前面に直接アクセスすることが必要である。

【1073】

不揮発性メモリの直接視

チップが、フラッシュメモリのフローティングゲートを放電させることなく露出するためにスライスされているならば、鍵は、おそらく、STM又はSKM（走査ケルビン顕微鏡

50

）を使用して直接監視できる。しかし、ゲートを放電させることなくこのレベルでチップをスライスすることはおそらく実施不可能である。湿式エッチング、プラズマエッチング、イオンミリング（集光イオンビームエッチング）、又は化学機械研磨は、ほぼ確実にフローティングゲート上に存在する小さい電荷を放出する。

【1074】

状態変化によって生じる光バーストの監視

ゲートの状態が変化するとき、少量の赤外線エネルギーが放出される。シリコンは赤外線透過性であるため、これらの変化はチップの下側から回路を注視することによって観測できる。放出プロセスは弱い、天文学で使用するために開発された超高感度機器を用いて検出できる程度に輝いている。IBMによって開発されたこの技術は、PICA（ピコセコンド・イメージング・サーキット・アナライザ）と呼ばれる。レジスタの状態が時点 t で既知であるならば、レジスタ切り替わり時間の監視によって、時点 $t+n$ における正確な値が明らかになり、データが鍵の一部であるならば、その部分は解読される。

10

【1075】

EMIの監視

電子回路が動作しているときには、かならず微弱な電磁信号が放出される。比較的低価格（数千ドル）装置はこれらの信号を監視できる。これは、攻撃者が鍵を推測するために十分な情報を与える。

【1076】

I_{dd} 変動の監視

鍵を監視できないとしても、レジスタが状態を変化させるときにはかならず電流の変動が生じる。信号対雑音比が十分に高い場合、攻撃者は、ハイビット又はロービットをプログラミングするときに生じる I_{dd} の差を監視することができる。 I_{dd} の変化は鍵に関する情報を示し得る。このような攻撃は、スマートカードを解読するために既に使用されている。

20

【1077】

差分故障解析

この攻撃は、イオン化、マイクロ波放射、又は環境ストレスによってビット誤りが取り込まれることを前提としている。殆どのケースでは、このような誤りは、鍵を明らかにするであろう役に立つ変更ではなく、チップに悪影響を与える（例えば、プログラムコードを破壊させる）可能性が高い。ROM上書き、ゲート破壊等のような目標としている故障は、有用な結果を生じる可能性が更に低い。

30

【1078】

クロック誤作動攻撃

チップは、典型的に、一定のクロック速度レンジ内で適切に動作するように設計されている。一部の攻撃者は、チップを非常に高いクロック速度で動作させることによってロジックに故障を生じさせ、又は特定の間隔で特定の時間にクロック誤作動を生じさせることを試みる。このアイデアは、回路が適切に機能しない乱調状態を作ることである。一例として、ANDゲートは、（乱調状態のために）入力 $_1$ と入力 $_2$ の論理積ではなく、常に入力 $_1$ をスルーする。攻撃者がチップの内部構造を知っているならば、攻撃者は、アルゴリズム実行中の正確な時点で乱調状態を取り入れ、これにより、鍵に関する情報を明らかにすることができる（或いは、最悪ケースでは、鍵自体を明らかにする）。

40

【1079】

電源攻撃

クロック信号に誤動作を引き起こす代わりに、攻撃者は電源に誤動作を生じさせることができ、電力は動作電圧レンジを外れるように増加又は減少させられる。正味の効果はクロック誤動作の場合と同じであり、特定の命令の実行に誤りを生じさせる。このアイデアは、CPUが鍵をXOR演算することを停止させること、又はデータを1ビット位置だけシフトすることを停止させること等である。鍵に関する情報を明らかにするために、特定の命令が目標にされる。

50

【1080】

ROMの上書き

ROMの単一ビットは、レーザーカッター顕微鏡を使用して、ロジックの向きに応じて、1又は0に書き換えることができる。あるオペコード／オペランドのセットがある場合、攻撃者が条件付きジャンプを無条件ジャンプへ書き換えること、又はたぶんレジスタ転送の宛先を変更することは簡単なことである。目標命令が慎重に選択されたならば、鍵が明らかにされるであろう。

【1081】

EEPROM／フラッシュの変更

EEPROM／フラッシュ攻撃は、ROM攻撃と類似しているが、レーザーカッター顕微鏡技術が個別のビットのセットとリセットの両方に使用できる点で異なる。これは、アルゴリズムの変更の適用範囲を非常に拡大する。

10

【1082】

ゲート破壊

AndersonとKuhnは、BihamとShamirがDESに関する攻撃を発表した高速ソフトウェア暗号化に関する1997年のワークショップのランブセッションで、ゲート破壊を説明している。この攻撃は、従来のブロック暗号(DES)のハードウェア実装内の個々のゲートを破壊するためレーザーカッターを使用するものであった。この攻撃の正味の効果は、レジスタの特定のビットを強制的に「無価値」にすることである。BihamとShamirは、特定のレジスタに強制的にこのような影響を与えることの効果一丸め関数からの出力の最下位ビットが0にセットされる、を説明している。左半分と右半分の最下位6ビットを比較することにより、鍵の数ビットを復元することができる。このようにして多数のチップにダメージを加えると、完全な鍵の復元を容易に行うために、鍵に関する十分な情報を明らかにすることができる。このように変更された暗号化チップは、暗号化と復号化はもはや逆の関係ではなくなるという性質をもつであろう。

20

【1083】

上書き攻撃

フラッシュメモリを読み出す代わりに、攻撃者は、単に、レーザーカッター顕微鏡を用いて単一ビットをセットする。攻撃者が前の値を知らないとしても、新しい値がわかる。チップが動作し続けるならば、そのビットの元の状態は、新しい状態と同じであるはずである。チップが動作しなくなった場合、そのビットの元の状態は、現在の状態の論理否定NOTである。攻撃者は、鍵の各ビットに対してこの攻撃を実行し、最大でnチップを使用してnビット鍵を獲得することができる(新しいビットが古いビットと一致するならば、新しいビットは次のビットを決定するために必要ではない)。

30

【1084】

テスト回路攻撃

殆どの回路は、製造上の欠陥を検査するために特に設計されたテスト回路を備えている。これは、BIST(ビルト・イン・セルフ・テスト)とスキャンパスを含む。非常に頻繁に、スキャンパス及びテスト回路は、内蔵された全てのラッチへのアクセス及びリードアウト機構を含む。一部のケースでは、テスト回路は特定のレジスタの内容に関する情報を与えるために使用される可能性がある。テスト回路は、しばしば、チップが全ての製造上のテストに合格すると、一部のケースではチップ内の特定の配線を切ることによって、無効状態にされる。しかし、やる気のある攻撃者は、テスト回路を再接続して、動作可能状態にし得る。

40

【1085】

残留メモリ

値は、電源が取り外された後、長期間RAMに残存するが、不揮発性とみなされるまで長期間残存することはない。攻撃者は、機密情報がRAM(例えば、作業用レジスタ)へ移された後、電源を取り外し、RAMから値を読み出そうとすることが可能である。この攻撃は、通常のRAMチップを有するセキュリティシステムに対して最も有効である。典型

50

的な例として、セキュリティシステムは、コンピュータケースが開けられたときにトリガーされる自動電源遮断機能付きで設計されていた。攻撃者は、単にケースを開け、RAMチップを取り外し、メモリ残留を利用して鍵を取り出せばよかった。

【1086】

チップ窃盗攻撃

認証チップの耐用期間に多数のステージが存在する場合、これらのステージの各々は、チップが盗み取られた場合のセキュリティ用の派生問題の観点で調べる必要がある。例えば、情報が複数のステージでチップにプログラミングされる場合、ステージ間でチップを窃盗したならば、攻撃者は鍵情報を入手できるか、又は攻撃のための労力が軽減される可能性がある。同様に、チップが、製造直後であり、プログラミングの前に盗み取られた場合、攻撃者に論理的利益又は物理的利益があるだろうか？

10

必要条件

消耗品を認証する問題に対する従来のソリューションは、典型的に、物理的なパッケージングについての特許に依存している。しかし、これは、家庭での詰め替え作業や産業財産権の保護の弱い国における複製品製造を止められない。その結果、より高レベルの保護が必要である。したがって、認証メカニズムは、システムが消耗品を安全かつ容易に認証できるようにする認証チップ53内に構築される。ここでは、消耗品を認証するシステムだけに限定すると（システムを認証する消耗品については考えない）、次の2レベルの保護を考えることができる：

プレゼンスオンリー認証

20

この場合、認証チップが存在することだけがテストされる。認証チップは、最プログラミングすることなく、別の消耗品で再使用することができる；

消耗品ライフタイム認証

この場合、認証チップの存在がテストされるだけでなく、認証チップ53は、消耗品の耐用期間に限り持続しなければならない。チップを再使用する場合、チップは、完全に消去され、再プログラミングされなければならない。これらの二つの保護レベルは、異なる必要条件を取り扱う。ここでは、主に、大量生産消耗品の複製品を防止するため、消耗品ライフタイム認証に関心がある。このケースでは、各チップは、認証対象である消耗品に関するセキュア状態情報を保持しなければならない。尚、消耗品ライフタイム認証チップは、プレゼンスオンリー認証チップを必要とするあらゆる状況で使用できることに注意する必要がある。認証の必要条件、データ記憶完全性の必要条件、及び製造の必要条件は、別々に検討すべきである。以下のセクションは各々の必要条件を簡単に説明する。

30

【1087】

認証

プレゼンスオンリー認証と消耗品ライフタイム認証の両方に対する認証必要条件は、システムが消耗品を認証するケースに限定される。プレゼンスオンリー認証の場合、認証チップが物理的に存在することが保証されなければならない。消耗品ライフタイム認証の場合、状態データが実際に認証チップに由来すること、及び状態データが途中で改ざんされていないこと、が保証される必要がある。これらの問題は切り離すことができず、改ざんされたデータは新しい送信元からのものであり、送信元を判定できない場合、改ざんの問題を解決できない。機密性があり、セキュリティ専門家によって精査されていない自家製のセキュリティ法に基づく認証方法を提供するだけでは不十分である。したがって、主要な必要条件は、専門家の精査に耐えた手段による認証を提供することである。認証チップ53によって使用される認証スキームは、論理的手段による打破に抵抗できなければならない。論理的タイプの攻撃は費用が掛かり、次の三つの事柄：

40

認証プロセスを完全に回避する；

あらゆる質問に回答できるように、力づく又は推論によって秘密鍵を取得する；

鍵を用いることなく各質問に正しい回答を与えるため、認証用の質問及び回答の性質を十分に検出する；

のうちの一つを実行しようとする。

50

【1088】

データ記憶完全性

認証プロトコルは、通信されるメッセージ中のデータ完全性の保証に対押しているが、データ記憶完全性も必要である。次の2種類のデータ：

秘密鍵のような認証データ；

シリアル番号及びメディア残量等の消耗品状態データ；

を認証チップ内に記憶しなければならない。

【1089】

これらの二つのデータタイプのアクセス必要条件は全く異なる。したがって、認証チップ53は、各タイプの完全性必要条件を考慮する記憶／アクセス制御メカニズムを必要とする。

10

【1090】

認証データ

認証データは機密保持されなければならない。認証データは、チップの耐用期間の製造／プログラミングステージの間にチップに記憶される必要があるが、それ以降、チップから取り出されることは許可されるべきではない。認証データは、不揮発性メモリから読み出されることに対抗できなければならない。認証スキームは鍵が推論によって取得できないことを保証する役目があり、製造プロセスは鍵が物理的手段によって取得できないことを保証する役目がある。認証データ記憶エリアのサイズは、必要な鍵、及び認証プロトコルによって義務づけられるような秘密情報を保持するために十分な大きさでなければならない。

20

【1091】

消耗品状態データ

各認証チップ53は、消耗品状態データのうちの256ビット（32バイト）を記憶できる。消耗品状態データは以下のタイプに分類される。アプリケーションに依存して、以下のタイプのデータ項目の各々の数は異なる。単一のデータ項目に対する最大数は32ビットであると仮定する：

リードオンリー；

リードライト；

デクリメントオンリー。

30

【1092】

リードオンリーデータは、チップの耐用期間の製造／プログラミングステージの間にチップに保存できなければならないが、それ以降、変更を許されるべきではない。リードオンリーデータの例は、消耗品バッチ番号及びシリアル番号である。

【1093】

リードライトデータは、変更可能な状態情報であり、例えば、特定の消耗品が使用された最後の時間である。リードライトデータ項目は、消耗品の耐用期間中、無制限の回数で読み書きをすることができる。それらは、消耗品に関するあらゆる状態情報を記憶するため使用できる。このデータに対する唯一の必要条件は、不揮発性メモリに保持されるべきである、ということである。攻撃者はシステムへのアクセス権を獲得し得るので（リードライトデータへの書き込みが可能であり）、攻撃者はこのタイプのデータフィールドを変更する可能性がある。このデータタイプは秘密情報のために使用されるべきではなく、不確かであると考えべきである。

40

【1094】

デクリメントオンリーデータは、消耗品リソースの利用可能度をカウントダウンするため使用される。例えば、コピー機のトナーカートリッジは、デクリメントオンリーデータ項目としてトナー残量を記憶する。カラープリンタ用のインクカートリッジは、各インク色の量をデクリメントオンリーデータ項目として記憶し、3個（シアン、マゼンタ及びイエローの各々に1個ずつ）、又は5から6個のデクリメントオンリーデータ項目を必要とする。この種のデータ項目の必要条件は、一旦、製造／プログラミングステージで初期値を

50

用いてプログラミングされると、値を減少させることだけが可能である、ということである。それは、最小値に達すると、それ以上は減少し得ない。デクリメントオンリーデータ項目は、消耗品ライフタイム認証だけで必要になる。

【1095】

製造

認証チップ53は、理想的には、低価格消耗品の認証メカニズムとして組み込むことができるように低製造コストであるべきである。認証チップ53は、フラッシュのような標準的な製造プロセスを使用すべきである。これは：

製造場所の非常に広範囲の選択を可能にさせること；

明確であり、且つ正常に動作するテクノロジーを使用すること；

コストを低減すること；

が必要である。

【1096】

使用される認証スキームとは無関係に、チップの認証部の回路は物理的攻撃に対抗できなければならない。物理的な攻撃は、以下の4つの主要な形で現れるが、攻撃の形態は変化する：

認証チップを完全に回避する；

動作中にチップを物理的に検査する（破壊的及び非破壊的）；

チップを物理的に分解する；

チップを物理的に改変する。

【1097】

理想的には、チップは米国から持ち出せるべきであるので、認証チップ53をセキュア暗号化装置として使用可能であってはならない。これは優先度の低い必要条件である。なぜならば、他の国にも認証チップを製造し得る多数の企業が存在するからである。いずれにせよ、米国からの国外持ち出し規制は変わり得る。

【1098】

認証

消耗品を認証する問題に対する従来のソリューションは、典型的に、物理的なパッケージングについての特許に依存している。しかし、これは、家庭での詰め替え作業や産業財産権の保護の弱い国における複製品製造を止められない。したがって、より高レベルの保護が必要である。機密性があり、セキュリティ専門家によって精査されていない自家製のセキュリティ法に基づく認証方法を提供するだけでは不十分である。ネットスケープの最初の独自に開発したシステム、及び携帯電話で私用されるGSM不正防止ネットワークは、設計秘密主義によって生じたセキュリティの脆弱性である。両方のセキュリティシステムは、それらの企業がオープン設計プロセスに従っていたならば検出できたであろう従来の手段によって破られた。ソリューションは、専門家の精査に耐えた手段による認証を提供することである。多数のプロトコルを消耗品認証のために使用することが可能である。ここでは、公表されたセキュリティ方法だけを使用し、この新しい方法によって知られた動作を使用する。全てのプロトコルにおいて、スキームの安全性は、秘密アルゴリズムではなく、秘密鍵に依拠する。プロトコルの全ては、時変性チャレンジ（即ち、毎回異なるチャレンジ）に依存し、レスポンスはチャレンジと秘密情報に依存する。チャレンジは乱数を含むので、観測者は後続のアイデンティフィケーションに関する有益な情報を集めることができない。プレゼンスオンリー認証と消耗品ライフタイム認証の各々のために二つのプロトコルが提案される。プロトコルは認証プロセスのために必要な認証チップの個数が異なるが、全てのケースで、システムは消耗品を認証する。ある種のプロトコルは、1個又は2個のチップで動作するが、他のプロトコルは2個のチップだけで動作する。使用される認証チップが1個であるか2個であるかは無関係に、システムは認証判定を行う役割を担う。

【1099】

シングルチップ認証

唯一の認証チップ 53 が認証プロトコルのために使用される場合、単一チップ（チップ A と呼ぶ）は、システム（システムと呼ぶ）に対し、チップ A が真正であることを伝える責任がある。プロトコルの開始時に、システムは、チップ A の真正性を確信できない。システムは、チップ A とチャレンジレスポンスプロトコルを実行し、チップの真正性を判定する。全てのプロトコルにおいて、消耗品の真正性は、チップの真正性に直接的に基づいている。即ち、チップ A が真正であるとみなされるならば、消耗品も真正であるとみなされる。データフローは図 167 に示されている。シングルチップ認証プロトコルの場合、システムは、ソフトウェア、ハードウェア、又は両者の組み合わせのどれでもよい。重要なことは、システムは安全ではなく、攻撃者が ROM を調べることによって、又は回路を調べることによって容易にリバースエンジニアリングをすることができる、ということである。システムは、それ自体で安全になるようには特に設計されていない。

10

【1100】

ダブルチップ認証

別のプロトコルでは、2 個の認証チップが図 168 に示されるように必要になる。単一のチップ（チップ A）は、システム（システムと呼ぶ）に対し、チップ A が真正であることを伝える責任がある。認証プロセスの一部として、システムは、信頼できる認証チップ（チップ T と呼ぶ）を利用する。ダブルチップ認証プロトコルでは、システムは、ソフトウェア、ハードウェア、又は両者の組み合わせのどれでもよい。しかし、チップ T は物理的な認証チップでなければならない。あるプロトコルでは、チップ T 及びチップ A は同じ内部構造を備え、他のプロトコルでは、チップ T 及びチップ A は異なる内部構造を備えている。

20

【1101】

プレゼンスオンリー認証（不確かな状態データ）

このレベルの消耗品認証の場合、認証チップ 53 の存在を確認することだけに関心がある。認証チップは状態情報を格納し得るが、その状態情報の伝送は安全であるとは考えられない。二つのプロトコルが提案される。プロトコル 1 は 2 個の認証チップが必要であり、プロトコル 2 は 1 個又は 2 個の認証チップを使用して実現可能である。

【1102】

プロトコル 1

プロトコル 1 はダブルチッププロトコルである（2 個の認証チップが必要になる）。各認証チップは以下の値を格納する：

30

$K_{F_K}[X]$ 用の鍵。秘密でなければならない；

R 現在の乱数。秘密でなくてもよいが、チップインスタンス毎に異なる初期値でシードされなければならない。Random 関数の呼び出し毎に変化する。

【1103】

各認証チップは以下の論理関数を格納する：

Random [] R を返し、R をシーケンス内で次へ進める；

F [X] 一方向関数 F を秘密鍵 K に基づいて X に適用した結果である $F_K[X]$ を返す。

【1104】

プロトコルは以下の通りである：

40

システムはチップ T からの Random [] を要求する；

チップ T は R をシステムへ返す；

システムはチップ T 及びチップ A の両方からの F [R] を要求する；

チップ T は $F_{K_T}[R]$ をシステムへ返す；

チップ A は $F_{K_A}[R]$ をシステムへ返す；

システムは、 $F_{K_T}[R]$ を $F_{K_A}[R]$ と比較する。それらが一致しているならば、チップ A は有効であるとみなされる。一致しないならば、チップ A は無効であるとみなされる。

【1105】

50

データフローは図 1 6 9 に示されている。システムは、 $F_K[R]$ メッセージを理解する必要がない。システムは、チップ A からの応答とチップ T からの応答が同じであることをチェックするだけでよい。したがって、システムは鍵を必要としない。プロトコル 1 の安全性は以下の 2 カ所にある：

$F[X]$ の安全性。認証チップだけが秘密鍵を格納するので、信頼できる認証チップ 5 3 (チップ T) によって生成された $F[X]$ と一致する $F[X]$ を X から生成し得るチップは真正に違いない；

全ての認証チップによって生成された R のドメインは大きく、非決定論的である筈である。全ての認証チップによって生成された R のドメインが小さいならば、複製品製造者は鍵を解読しなくてもよい。その代わりに、複製品製造者は、システムによって送信されたコードに対する純正チップからのレスポンスの全てが記録された ROM を自分のチップに組み込むことができる。Random 関数は厳密に認証チップ内に収容しなくてもよい。なぜならば、システムは、同じ乱数シーケンスを生成する可能性を秘めているからである。しかし、それは、システムの設計を単純化し、乱数生成器の安全性が認証チップを使用する全ての実装形態に対して同一であり、システム実装においてエラーが起こる可能性を減少させる。

【1 1 0 6】

プロトコル 1 には複数の利点がある：

K は認証プロセス中に明らかにされることがない；

X が与えられたとき、複製チップは、K を用いないで、又は実際の認証チップへアクセスしないで、 $F_K[X]$ を生成できない；

システムは、特に、インクジェットプリンタのような低価格システムにおいて容易に設計できる。なぜならば、暗号化又は復号化がシステム自体によって要求されないからである；

広範囲の鍵付き一方向関数が存在し、対称暗号化、乱数シーケンス、及びメッセージ認証コードが含まれる；

一方向関数は必要なゲート数が少なく、非対称アルゴリズムよりも簡単に検証することができる；

鍵付き一方向関数のための安全な鍵サイズは、非対称 (公開鍵) アルゴリズムの場合と同程度に大きくする必要がない；

$F[X]$ が対称暗号化関数であるならば、最小限の 1 2 8 ビットは、適切な安全性を提供することができる。

【1 1 0 7】

しかし、このプロトコルには問題がある：

これは選択テキスト攻撃の対象になりやすい。攻撃者は、チップを自分専用のシステムにプラグインし、選択された R を生成、出力を観測することができる。鍵を見つけるため、攻撃者は、特定の $F[M]$ を生成する R を探索することも可能である。なぜならば、多数の認証チップを並列にテストできるからである；

選択された一方向関数に応じて、鍵生成は複雑化し得る。よい鍵を選択する方法は、使用されるアルゴリズムに依存する。ある種の鍵は一定のアルゴリズムに対して弱い；

鍵付き一方向関数の選択自体は重要ではない。特許保護に由来するライセンスを必要とする場合もある。

【1 1 0 8】

媒介物は、平文メッセージ M をチップ A へ渡す前に平文メッセージ M に作用することが可能であり、これは、チップ A が M を見るまで媒介物が M を見ない場合には好ましい。媒介物が M を全く見ない場合にはより好ましい。

【1 1 0 9】

F が対称暗号化である場合、適切なセキュリティのために必要な鍵サイズのため、チップは米国から国外へ持ち出せない。なぜならば、チップは強い暗号化装置として使用できるからである。

【1110】

プロトコルが非対称暗号アルゴリズムとしてFを用いて実現されるならば、対称の場合を上回る利点は無く、鍵をより長くする必要があり、シリコン内の暗号化アルゴリズムがより高価になる。プロトコル1は、鍵の安全性を維持するため、2個の認証チップを用いて実現しなければならない。すなわち、各システムは認証チップが必要であり、各消耗品は認証チップが必要である。

【1111】

プロトコル2

あるケースでは、システムは大規模の処理能力を備えている。或いは、例えば、大量に製造されたシステムの場合、チップTをシステムに統合することは望ましい。非対称暗号アルゴリズムの使用によって、システムのチップT部を不確かにするのが許容される。したがって、プロトコル2は非対称暗号方式を使用する。このプロトコルの場合、各チップは以下の値を格納する：

$K_{E_K}[X]$ 及び $D_K[X]$ 用の鍵である。チップA内で秘密にされるべきである。チップT内で秘密にする必要がない；

R 現在の乱数。秘密でなくてもよいが、チップインスタンス毎に異なる初期値でシードされなければならない。Random関数の呼び出し毎に変化する。

【1112】

以下の関数が定義される：

$E[X]$ チップTだけ。 $E_K[X]$ を返す。但し、Eは非対称暗号化関数Eである；

$D[X]$ チップAだけ。 $D_K[X]$ を返す。但し、Dは非対称暗号化関数Dである；

Random[] チップTだけ。 $R \parallel E_K[R]$ を返す。RはシードSに基づく乱数である。乱数シーケンス内でRを次へ進める。

【1113】

公開鍵 K_T はチップT内にあり、一方、秘密鍵 K_A はチップAにある。 K_T がチップTにあることは、チップTをソフトウェア又はハードウェアで実現できるという利点がある（但し、Rのシードはチップ又はシステム毎に異なる）。したがって、プロトコル2は、シングルチッププロトコルとしてもダブルチッププロトコルとしても実現できる。認証のためのプロトコルは以下の通りである：

システムはチップTのRandom関数を呼び出す；

チップTは $R \parallel E_{K_T}[R]$ をシステムへ返す；

システムはチップAの関数Dを呼び出し、 $E_{K_T}[R]$ を渡す；

チップAは、 $D_{K_A}[E_{K_T}[R]]$ によって獲得されたRを返す；

システムは、チップAからのRとチップTによって生成された元のRを比較する。それらが一致しているならば、チップAは有効であるとみなされる。一致しないならば、チップAは無効であるとみなされる。

【1114】

データフローは図170に示されている。プロトコル2には次の利点がある：

K_A （秘密鍵）は認証プロセス中に明らかにされることがない；

$E_{K_T}[X]$ が与えられたとき、複製チップは、 K_A を用いないで、又は実際のチップAへアクセスすることなく、Xを生成できない；

$K_T \neq K_A$ であるため、チップTは、完全にソフトウェアで、不確かなハードウェアで、又はシステムの一部として実現できる。チップA（消耗品内）だけが安全な認証チップであることを要求される；

チップTが物理的なチップであるならば、システムは容易に設計できる；

多数の十分に文書化され暗号解読された非対称アルゴリズムの中から、特許が開放されライセンスフリーであるソリューションを含む、インプリメンテーション用のアルゴリズムを選択することができる。

【1115】

しかし、プロトコル2には多数の固有の問題がある：

十分な安全性のため、各鍵は2048ビットにする必要がある（これに対して、プロトコル1の対称暗号方式の場合、最小値は128ビットである）。暗号化及び復号化アルゴリズムによって使用される関連した中間メモリは、相応に大きくなる；

鍵生成は重要である。乱数は優れた鍵ではない；

チップTがコアとして実装された場合、チップTを所与のシステムASICに連結することが難しい；

チップTがソフトウェアとして実装された場合、システムの実装がプログラミングエラー及び厳密ではないテストに対して無防備であるだけではなく、コンパイラ及び数学的プリミティブの完全性をシステムのインプリメンテーション毎に厳密にチェックしなければならない。これは、単純に十分にテストされたチップを使用する場合よりも複雑であり、且つコスト高である；

多数の対称アルゴリズムが特に差分暗号解読（選択テキスト攻撃に基づいている）に対抗するため強化されているにもかかわらず、秘密鍵 K_A は選択テキスト攻撃の対象になりやすい；

チップA及びチップTが同じ認証チップのインスタンスである場合、各チップは、非対称暗号化及び復号化の両方の機能を備える必要がある。その結果として、各チップは、プロトコル1のために必要なチップよりも大型化し、複雑化し、高価になる；

認証チップが、コストを節約し、設計／テストの複雑さを低減するため2個のチップに分割された場合、2個のチップを依然として製造する必要がある、スケールメリットが減少する。これは、システムの消耗品に対する相対的な個数によって相殺されるが、依然として考慮することが必要である；

プロトコル2の認証チップは、米国から国外へ持ち出すことができない。なぜならば、それらは、強力な暗号化装置であるとみなされるからである。

【1116】

たとえば、プロトコル2用の鍵を選択するプロセスが簡単であるとしても、プロトコル2は、（鍵サイズ及び機能的実装の両方の）シリコン実装のコストが高いため、現時点では実際的ではない。したがって、プロトコル1がプレゼンスオンリー認証のためのプロトコルとして選択される。

【1117】

本物の認証チップを使用する複製消耗品

プロトコル1及び2は、チップAが本物の認証チップであるかどうかをチェックするだけである。それらは、消耗品自体が有効であるかどうかを調べるためのチェックを行わない。認証に関する基本的な前提として、チップAが有効であるならば、消耗品は有効である。したがって、複製品製造者は本物の認証チップを複製消耗品に挿入することが可能である。以下の二つのケースを検討する：

状態データが認証チップに書き込まれないケースでは、チップは完全に再使用可能である。したがって、複製品製造者は、有効な消耗品を複製消耗品にリサイクルすることができる。これは、認証チップを消耗品の物理的パッケージングに融合することによって困難になるが、詰め替え作業者を止められないであろう；

状態データが認証チップに書き込まれるケースでは、チップは、新品か、部分的に使用されているか、又は完全に使い切られている。しかし、これは、複製品製造者によるピギーバック攻撃の使用を止められず、複製品製造者はピギーバックとして本物の認証チップを具備したチップを構築する。攻撃者のチップ（チップE）は、媒介物である。電源投入時、チップEは、本物の認証チップ53から固有のメモリへ全ての記憶状態値を読み込む。チップEは、次に、システムからの要求を検査し、要求に応じて様々な動作をとる。認証要求は、本物の認証チップ53へそのまま渡すことができ、一方、リード／ライト要求は、本物の認証チップの動作と似ているメモリによってシミュレーションされ得る。このようにして、認証チップ53は、常に、電源投入時に新品であるかのように見える。チップEがこれを行うことができるのは、データアクセスは認証されないからである。

【1118】

システムを欺いて、システムのデータアクセスが成功したと思わせるために、チップEは、依然として本物の認証チップを必要とし、第2のケースでは、複製チップが本物のチップの他に必要になる。その結果として、プロトコル1及び2が有効になるのは、複製品製造者が本物の認証チップ53を消耗品に埋め込むことの費用対効果がよくない場合である。消耗品をリサイクルできない、又は簡単に詰め替えできないならば、プロトコル1又はプロトコル2の使用によって十分な保護が得られる。複製動作を成功させるためには、各複製消耗品は、有効な認証チップを具備しなければならない。そのチップは、まとめて盗み取るか、又は古い消耗品から取り出さなければならない。これらの再生チップの量は（それらを再生する労力と共に）ビジネスの基盤とするためには不十分であり、セキュアデータ転送（プロトコル3及び4を参照）の保護を追加する実益はない。

10

【1119】

鍵の寿命

上記の二つのプロトコルの一般的な問題は、一旦、認証鍵が選択されると、それを変更することが容易ではない点である。ある種の状況では、鍵が危険にさらされることは問題ではないが、他の場合には、鍵を危うくすることは致命的である。例えば、車／車キーシステム／消耗品のシナリオでは、消費者は、1組の車／車キーだけを保有する。各車は異なる認証鍵を有する。その結果として、車キーの損失は個々の車を危うくするだけである。所有者がこのことを問題であると考えるならば、所有者は、車のエレクトロニクス内のシステムチップを交換することによって、新しい「車キー」を入手しなければならない。所有者の鍵は、新しい「車のシステム認証チップ」と連動するように再プログラミング／交換する必要がある。これに対して、大量消耗品市場（例えば、プリンタのインクカートリッジ）の鍵が危うくなることは、複製インクカートリッジ製造者が自分専用の認証チップを製造可能にさせる。既存のシステムに対する唯一のソリューションはシステム認証チップを更新することであり、これは、コストが掛かり、物流的に困難な課題である。いずれにしても、消費者のシステムは既に正常に機能し、既存の機器を妨害しようとする動機がない。

20

【1120】

消耗品ライフタイム認証

この消耗品認証のレベルでは、認証チップの存在を確認すること、並びに、認証チップが消耗品と同じ期間だけ確実に存続することに関心がある。認証チップが存在することを確認することに加えて、認証チップのメモリ空間の書き込み及び読み出しも同様に認証されなければならない。このセクションでは、認証チップのデータ記憶完全性は安全であると仮定し、メモリの一部はリードオンリーであり、他の一部はリード／ライトであり、その他はデクリメントオンリーである（より詳しくは、データ記憶完全性というタイトルの章を参照のこと）。二つのプロトコルが存在する。プロトコル3は2個の認証チップを必要とし、プロトコル4は1個又は2個の認証チップを使用して実現することができる。

30

【1121】

プロトコル3

このプロトコルはダブルチッププロトコルである（2個の認証チップが必要になる）。このプロトコルの場合、各認証チップは以下の値を格納する：

40

K_1 $F_{K_1}[X]$ を計算するための鍵。秘密にしなければならない；

K_2 $F_{K_2}[X]$ を計算するための鍵。秘密にしなければならない；

R 現在の乱数。秘密でなくてもよいが、チップインスタンス毎に異なる初期値でシードされなければならない。Test関数によって定義されるように認証が成功する毎に変化する；

M 認証チップ53のメモリベクトル。この空間の一部はチップ毎に異なるべきである（乱数でなくてもよい）。

【1122】

各認証チップは以下の論理関数を格納する：

F[X] 内部関数専用。一方向関数Fを鍵 K_1 又は鍵 K_2 の何れかに基づいてXに適用

50

した結果である $F_K[X]$ を返す；

$Random[]$ $R \mid F_{K_1}[R]$ を返す；

$Test[X, Y]$ $F_{K_2}[R \mid X] = Y$ であるならば、1 を返し R を進める。そうでなければ、0 を返す。0 を返すために要する時間は、全ての不良な入力に対して同じでなければならない；

$Read[X, Y]$ $F_{K_1}[X] = Y$ であるならば、 $M \mid F_{K_2}[X \mid M]$ を返す。そうでなければ、0 を返す。0 を返すために要する時間は、全ての不良な入力に対して同じでなければならない；

$Write[X]$ M のうち正当に書き換え可能な部分に X を書き込む。

【1123】

10

チップ A を認証し、チップ A のメモリ M を読み出すためには、

システムはチップ T の $Random$ 関数を呼び出し、

チップ T は $R \mid F_K[R]$ を生成し、これをシステムへ返し、

システムはチップ A の $Read$ 関数を呼び出し、 R 、 $F_K[R]$ を渡し、

チップ A は M 及び $F_K[R \mid M]$ を返し、

システムはチップ T の $Test$ 関数を呼び出し、 M 及び $F_K[R \mid M]$ を渡し、

システムはチップ T からのレスポンスをチェックする。レスポンスが 1 であるならば、チップ A は真正であるとみなされる。0 であるならば、チップ A は無効であるとみなされる。

【1124】

20

チップ A のメモリ M への M_{new} の書き込みを認証するためには、

システムはチップ A の $Write$ 関数を呼び出し、 M_{new} を渡し、

読み出しのための認証手続きが実行され、

チップ A が真正であり、 $M_{new} = M$ であるならば、書き込みは成功である。そうでなければ、書き込みは失敗した。

【1125】

読み出し認証のデータフローは図 171 に示されている。プロトコル 3 に関して最初に注意しなければならないことは、 $F_K[X]$ を直接呼び出せないことである。その代わりに、 $F_K[X]$ は、 $Random$ 、 $Test$ 及び $Read$ によって間接的に呼び出される：

$Random[]$ は $F_{K_1}[X]$ を呼び出す。X は呼び出し元によって選択されない。X は $Random$ 関数によって選択される。攻撃者は、希望の X 、 $F_{K_1}[X]$ のペアを獲得するためには、 $Random$ 、 $Read$ 、及び $Test$ の複数回の呼び出しを使用してブルートフォース検索を実行しなければならない；

$Test[X, Y]$ は $F_{K_2}[X]$ を呼び出す。結果をそのまま返すのではなく、結果を Y と比較し、1 又は 0 を返す。所定の X に対して $F_{K_2}[R \mid X]$ の様々な値を試して、 $Test$ を複数回呼び出すことによって K_2 を推論しようとする試みは、 R が攻撃者によって選択できないブルートフォース検索まで軽減される；

$Read[X, Y]$ は $F_{K_1}[X]$ を呼び出す。X 及び $F_{K_1}[X]$ は、呼び出し元によって供給されなければならないので、呼び出し元は、前もって X 、 $F_{K_1}[X]$ ペアが分かっている必要がある；

この呼び出しは、 $Y \neq F_{K_1}[X]$ であるならば 0 を返すので、呼び出し元は、 K_1 に対するブルートフォース攻撃のため $Read$ 関数を使用することができる；

$Read[X, Y]$ は $F_{K_2}[X \mid M]$ を呼び出す。X は呼び出し元によって供給されるが、X は $Random$ 関数によって既に与えられた値だけをとり得る（なぜならば、X 及び Y は K_1 によって検証されている）。このため、選択テキスト攻撃は、最初に、 $Random$ からペアを収集しなければならない（事実上、ブルートフォース攻撃である）。その上、 M の一部だけが選択テキスト攻撃で使用される。なぜならば、 M の一部は一定であり（リードオンリー）、 M のデクリメントオンリー部分は、消耗品毎に 1 回ずつ使用できる。次の消耗品で、 M のリードオンリー部分は異なる。

【1126】

50

$F_K[X]$ が間接的に呼び出されることは、認証チップへの選択テキスト攻撃を阻止する。攻撃者は、望ましい R が現れるまで、 $Random$ 、 $Read$ 、及び $Test$ を複数回呼び出すことによって選択された R 、 $F_{K_1}[R]$ ペアしか獲得できないので、 K_1 へのブルートフォース攻撃は、 K_2 への制限選択テキスト攻撃を実行するために必要である。 K_2 への選択テキスト攻撃における試みは制限されるであろう。なぜならば、テキストを完全に選択できないからであり、 M の一部はリードオンリーであり、依然として各認証チップに対して異なる。2 番目に注意すべきことは、2 個の鍵が使用されることである。小さいサイズの M が与えられた場合、2 種類の鍵 K_1 及び K_2 は、 $F[R]$ と $F[R|M]$ の間に相関が無いことを保証するため使用される。したがって、 K_1 は、差分攻撃から K_2 を保護するために使用される。1 個のより長い鍵を使用することは不十分である。なぜならば、 M は 256 ビットだけであり、 M の一部分しか消耗品の耐用期間中に変化しないからである。そうでなければ、攻撃者は、ある種の未発見の技術によって、限定された M の変化が、特定の R のビット組み合わせに与える影響を判定し、 $F_{K_1}[X]$ に基づいて $F_{K_2}[X|M]$ を計算できる可能性がある。付加的な事前計算として、チップ A の $Random$ 関数及び $Test$ 関数は無効にされるべきであり、その結果、 R 、 $F_K[R]$ のペアを生成するため、攻撃者は、チップ T のインスタンスを使用する必要がある、その各々はチップ A よりも高価である（なぜならば、システムは各チップ T に対して獲得されなければならないからである）。同様に、 $Random$ 、 $Read$ 及び $Test$ の呼び出しの間の遅れは最小限にされるべきであり、その結果として、攻撃者はこれらの関数を高速で呼び出すことができない。このようにして、各チップは、ある時間に、特定の個数の X 、 $F_K[X]$ のペアだけを明らかにする。プロトコル 3 の唯一の特有のタイミング必要条件は、返り値 0（不良な入力を示す）が、誤りが入力のごどこに存在するかとは無関係に、同じ時間で生成されるべきである、ということである。したがって、攻撃者は、入力値に関する誤りが何であるかに関して学習できない。これは、 RD 関数と TST 関数の両方に対しても成り立つ。

【1127】

プロトコル 3 に関して注意すべき別の事項は、チップ A からのデータ読みだしにもチップ A の認証が必要になる、ということである。システムは、 $F_{K_2}[R|M]$ が正しく返された場合に、メモリ (M) の内容はチップ A が主張している内容そのものであることを確認できる。複製チップは、 M がある値であると偽る（例えば、消耗品が一杯詰まっているように見せかける）かもしれないが、複製チップは、システムによって渡された任意の R に対して、 $F_{K_2}[R|M]$ を返すことができない。このように、事実上の署名、 $F_{K_2}[R|M]$ は、認証チップ A が M を送信したことだけではなく、 M がチップ A とシステムの間で改ざんされていないことをシステムに保証する。最後に、上述の $Write$ 関数は書き込みを認証しない。書き込みを認証するため、システムは、各書き込みの後に読み出しを実行しなければならない。プロトコル 3 には幾つかの基本的な利点がある：

K_1 及び K_2 は認証プロセス中に明らかにされることがない；

X が与えられたとき、複製チップは、 K を用いないで、又は本物のチップ A へアクセスすることなく、 $F_{K_2}[X|M]$ を生成できない；

システム自体によって暗号化又は復号化が必要ではないため、特に、インクジェットプリンタのような低コストシステムにおいて、システムは簡単に設計できる；

対称暗号方式、乱数シーケンス、及びメッセージ認証コードを含む広範囲の鍵ベースの一方方向関数が存在する；

鍵付き一方方向関数は必要なゲート数が少なく、非対称アルゴリズムよりも簡単に検証することができる。）；

鍵付き一方方向関数のための安全な鍵サイズは、非対称（公開鍵）アルゴリズムの場合と同程度に大きくする必要がない。 $F[X]$ が対称暗号化関数であるならば、最小限の 128 ビットで適切な安全性を提供することができる。

【1128】

したがって、プロトコル 3 の場合、チップ A を認証する唯一の方法は、チップ A のメモリ

の内容を読み出すことである。このプロトコルの安全性は、基本となる $F_K[X]$ スキームと、すべてのシステムの組に亘る R のドメインと、に依存する。 $F_K[X]$ は鍵付き一方向関数であればよいが、それを非対称暗号として実現することに利点はない。鍵はより長くすることが必要であり、暗号化アルゴリズムはシリコン内でより高価になる。このため、非対称アルゴリズムと共に使用するための第2のプロトコル、即ち、プロトコル4が生じる。プロトコル3は、鍵を秘密のまま保つために2個の認証チップで実現する必要がある。即ち、各システムは認証チップを必要とし、各消耗品は認証チップを必要とする。

【1129】

プロトコル4

あるケースでは、システムは大規模の処理能力を備えている。或いは、例えば、大量に製造されたシステムの場合、チップTをシステムに統合することは望ましい。非対称暗号アルゴリズムの使用によって、システムのチップT部を不確かにするのが許容される。したがって、プロトコル4は非対称暗号方式を使用する。このプロトコルの場合、各チップは以下の値を格納する：

K $E_K[X]$ 及び $D_K[X]$ 用の鍵である。チップA内で秘密にされるべきである。チップT内で秘密にする必要がない；

R 現在の乱数。秘密でなくてもよいが、チップインスタンス毎に異なる初期値でシードされなければならない。 $Test$ 関数によって定義されるように認証が成功する毎に変化する；

M 認証チップ53のメモリベクトル。この空間の一部はチップ毎に異なるべきである（乱数でなくてもよい）。

【1130】

$Read$ 関数で何かを検証する意味はない。なぜならば、誰でも公開鍵を使用して暗号化できるからである。したがって、以下の関数が定義される：

$E[X]$ 内部関数だけ。 $E_K[X]$ を返す。但し、 E は非対称暗号化関数 E である；

$D[X]$ 内部関数だけ。 $D_K[X]$ を返す。但し、 D は非対称暗号化関数 D である；

$Random[]$ チップTだけ。 $E_K[R]$ を返す；

$Test[X, Y]$ $D_K[R | X] = Y$ であるならば、1を返し R を進める。そうでなければ、0を返す。0を返すために要する時間は、全ての不良な入力に対して同じでなければならない；

$Read[X]$ $M | E_K[R | M]$ を返す。但し、 $R = D_K[X]$ である（入力をテストしない）；

$Write[X]$ M のうち正当に書き換え可能な部分に X を書き込む。

【1131】

公開鍵 K_T はチップT内にあり、一方、秘密鍵 K_A はチップAにある。 K_T がチップTにあることは、チップTをソフトウェア又はハードウェアで実現できるという利点がある（但し、 R はシステム毎に異なる乱数でシードされる）。チップAを認証し、チップAのメモリ M を読み出すためには、

システムはチップTの $Random$ 関数を呼び出し、

チップTは $E_{K_T}[R]$ を生成し、これをシステムへ返し、

システムはチップAの $Read$ 関数を呼び出し、 $E_{K_T}[R]$ を渡し、

チップAは $M | E_{K_A}[R | M]$ を返し、最初に $D_{K_A}[E_{K_T}[R]]$ によって R を獲得し、

システムはチップTの $Test$ 関数を呼び出し、 M 及び $E_{K_A}[R | M]$ を渡し、

チップTは $D_{K_T}[E_{K_A}[R | M]]$ を計算し、それを $R | M$ と比較し、

システムはチップTからのレスポンスをチェックする。レスポンスが1であるならば、チップAは真正であるとみなされる。0であるならば、チップAは無効であるとみなされる。

【1132】

チップAのメモリ M への M_{new} の書き込みを認証するためには、

システムはチップ A の $Write$ 関数を呼び出し、 M_{new} を渡し、読み出しのための認証手続きが実行され、チップ A が真正であり、 $M_{new} = M$ であるならば、書き込みは成功である。そうでなければ、書き込みは失敗した。

【1133】

読み出し認証のデータフローは図 172 に示されている。有効なチップ A だけに R の値がわかる。なぜならば、R は認証関数に渡されないからである（それは暗号化された値として渡される）。R は $E[R]$ を復号化することによって獲得しなければならず、これは、秘密鍵 K_A を使用することによってのみ行える。R が獲得されると、R は M に付加され、結果が再符号化される。チップ T は、復号化された形式の $E_{K_A}[R | M] = R | M$ を検証することができ、これにより、チップ A は有効である。 $K_T \neq K_A$ であるため、 $E_{K_T}[R] \neq E_{K_A}[R]$ である。プロトコル 4 には以下の利点がある：

K_A （秘密鍵）は認証プロセス中に明らかにされることがない；

$E_{K_T}[X]$ が与えられたとき、複製チップは、 K_A を用いないで、又は実際のチップ A へアクセスすることなく、X を生成できない；

$K_T \neq K_A$ であるため、チップ T は、完全にソフトウェアで、不確かなハードウェアで、又はシステムの一部として実現できる。チップ A だけが安全な認証チップであることを要求される；

チップ T 及びチップ A は異なる鍵を格納するので、チップ T の激しいテストによっても K_A に関して何も明らかにならない；

チップ T が物理的なチップであるならば、システムは容易に設計できる；

多数の十分に文書化され暗号解読された非対称アルゴリズムの中から、特許が開放されライセンスフリーであるソリューションを含む、インプリメンテーション用のアルゴリズムを選択することができる；

チップ A の要求がチップ T へ向けられるようにシステムを再配線できるとしても、 $K_T \neq K_A$ であるため、チップ T はチップ A に対して回答できない。攻撃は、認証プロトコルを回避するため、システム ROM 自体へ向けられるべきであろう。

【1134】

しかし、プロトコル 4 には多数の問題がある：

全ての認証チップは、非対称暗号化及び復号化の両方の機能を収容する必要がある。その結果として、各チップは、プロトコル 3 のために必要なチップよりも大型化し、複雑化し、高価格化する；

十分な安全性のため、各鍵は 2048 ビットにする必要がある（これに対して、プロトコル 1 の対称暗号方式の場合、最小値は 128 ビットである）。暗号化及び復号化アルゴリズムによって使用される関連した中間メモリは、相応に大きくなる；

鍵生成は重要である。乱数は優れた鍵ではない；

チップ T がコアとして実装された場合、チップ T を所与のシステム ASIC に連結することが難しい；

チップ T がソフトウェアとして実装された場合、システムの実装がプログラミングエラー及び厳密ではないテストに対して無防備であるだけでなく、コンパイラ及び数学的プリミティブの完全性をシステムのインプリメンテーション毎に厳密にチェックしなければならない。これは、単純に十分にテストされたチップを使用する場合よりも複雑であり、且つコスト高である；

多数の対称アルゴリズムが特に差分暗号解読（選択テキスト攻撃に基づいている）に対抗するため強化されているにもかかわらず、秘密鍵 K_A は選択テキスト攻撃の対象になりやすい。

【1135】

プロトコル 4 の認証チップは、米国から国外へ持ち出すことができない。なぜならば、それらは、強力な暗号化装置であるとみなされるからである。

【1136】

プロトコル 3 の場合と同様に、プロトコル 4 の唯一の特有のタイミング必要条件は、返り値 0（不良な入力を示す）が、誤りが入力のどこに存在するかとは無関係に、同じ時間で生成されるべきである、ということである。したがって、攻撃者は、入力値に関する誤りが何であるかに関して学習できない。これは、R D 関数と T S T 関数の両方に対しても成り立つ。

【1137】

T S T 呼び出しの変化

2 個の認証チップが使用される場合、理論的には、複製品製造者はシステム認証チップを、T S T の各呼び出しに 1（成功）を返すもので置換することが可能である。システムは、多数回、間違ったハッシュ値を用いて N 回に亘って T S T を呼び出すことによってこれをテストし、その結果が 0 であることを予測することができる。T S T が最後に呼び出されたとき、チップ A からの真の戻り値が渡され、その戻り値は信頼できる。T S T を呼び出す回数が問題になる。呼び出しの回数はランダムでなければならない。これにより、複製チップ製造者は前もって回数を知り得ない。システムがクロックを持っているならば、クロックからのビットは、間違った T S T 呼び出しを何回行わなければならないかを定めるために使用される。それ以外の場合、チップ A からの戻り値を使用できる。後者のケースでは、攻撃者は、依然として、複製チップ T がチップ A からの戻り値を監視できるようにするため、システムを書き換えることができ、これにより、正しいハッシュ値を知ることができる。最悪のケースは、勿論、システムが、認証された消耗品を必要としない複製システムによって完全に置き換えられた場合であり、これは、システムを書き換え、変更する限界のケースである。このため、T S T 呼び出しの変化はオプションであり、システム、消耗品、及び変更が行われる可能性に依存する。このようなロジックをシステムに追加することは、システムがより複雑になるので、（例えば、小型デスクトッププリンタの場合）それだけの価値があるとは考えられない。これに対して、このようなロジックをカメラに追加することは、それだけの価値があると考えられる。

【1138】

本物の認証チップを使用する複製消耗品

消耗品部分を使用する前に消耗品残量をデクリメントすることは重要である。消耗品が先に使用されると、複製消耗品は、特別の既知アドレスへの書き込みの間に、接触の不備のふりをして、新しい消耗品のように見えるからである。この攻撃は、依然として、各消耗品に本物の認証チップが必要であることが重要である。

【1139】

鍵の寿命

上記の二つのプロトコルの一般的な問題は、一旦、認証鍵が選択されると、それを変更することが容易ではない点である。ある種の状況では、鍵が危険にさらされることは問題ではないが、他の場合には、鍵を危うくすることは致命的である。

【1140】

プロトコルの選択

プロトコル 2 及び 4 の鍵の選択が簡単であるとしても、両方のプロトコルは、（鍵サイズと機能的な実装の両面で）シリコン実装のコストが高いため、現時点では実際的ではない。したがって、選択されるプロトコルは、プロトコル 1 及び 3 である。しかし、プロトコル 1 及び 3 は、多数の同じコンポーネントを含み、どちらも書き込み及び読み出しアクセスが必要であり、どちらも鍵付き一方向関数の実装が必要であり、どちらも乱数生成機能が必要である。

【1141】

プロトコル 3 は付加的な鍵（ K_2 ）が必要であると共に、ある種の最小限の状態機械の変化：

R a n d o m 中に $F_{K_1}[X]$ を呼び出せるような状態機械の変更と、
 $F_{K_2}[X]$ を呼び出す T e s t 関数と、

10

20

30

40

50

$F_{K_1}[X]$ 及び $F_{K_2}[X]$ を呼び出すために状態機械の Read 関数への変更と、が必要である。

【1142】

プロトコル 3 はプロトコル 1 に対して最小限の変更だけが必要である。プロトコル 3 の方が安全性が高く、プレゼンスオンリー認証（プロトコル 1）が要求されるような全ての状況で使用可能である。したがって、それは特に優れたプロトコルである。プロトコル 1 及び 3 の両方が鍵付き一方向関数を使用する場合に、一方向関数の選択をより詳細に説明する。以下の表は、適用可能な選択肢の属性の概要を示している。属性は、その属性が利点であることが分かるように表されている。

【1143】

【表 103】

	トリプル DES	ブローフィッシュ	RCS	IDEA	ランダムシーケンス	HMAC-MD5	HMAC-SHA1	HMAC-RIPEMD-160
特許開放								
ランダム鍵生成								
米国から持ち出し可能								
高 速								
本アプリケーションでの使用に好適な鍵サイズ (ビット)	168	128	128	128	512	128	160	160
ブロックサイズ (ビット)	64	64	64	64	256	512	512	512
暗号解読アタックフリー (弱い鍵は別)								
入力サイズ N に対する出力	$\geq N$	$\geq N$	$\geq N$	$\geq N$	128	128	160	160
低記憶必要量								
低シリコン複雑性								
NSA 設計								

この表を検討することによって、事実上、3 種類の HMAC 構造とランダムシーケンスとの間で選択されることがわかる。鍵サイズ及び鍵生成の問題は、ランダムシーケンスを除外する。多数の攻撃が既に MD5 に対して実行されていることを考えると、ハッシュ結果は 128 ビットに過ぎないので、HMAC-MD5 も除外される。したがって、HMAC-SHA1 と HMAC-RIPEMD160 の間で選択を行うことになる。RIPEMD-160 は比較的新しく、SHA1 のようには大規模に暗号解読されていない。しかし、SHA-1 は NSA によって設計されているので、これは、否定的な属性として見られることもある。

【1144】

両者の間に大差がないのであれば、SHA-1 が HMAC 構造用に使用されるであろう。

【1145】

乱数生成器の選択

上述の各プロトコル (1-4) は、乱数生成器を必要とする。生成器は、全てのシステムの耐用期間に亘って生成される乱数が予測不能であるという意味で「優良」でなければならない。乱数が各システムに対して同一であるならば、攻撃者は、簡単に、本物の認証チップからの正しいレスポンスを記録し、そのレスポンスを複製チップ用の ROM ルックアップに収容することが可能である。このような攻撃の場合、 K_1 又は K_2 を獲得しなくて

もよい。したがって、各システムからの乱数は、予測不能であるために、即ち、非決定論的であるために十分に異なることが必要である。このようにして、R（ランダムシード）の初期値は、特定のビットが1であるか、又は0であるかに関する情報が存在しない物理的にランダムな現象から収集された物理的に生成された乱数を用いてプログラミングされるべきである。Rのシードは、決して、コンピュータ上で実行される乱数生成器で生成してはならない。そうでなければ、生成器アルゴリズム及びシードが解読され、攻撃者は、全てのシステムの全てのRの組を生成し、知ることができるようになる。

【1146】

各認証チップのRシードが異なることは、最初のRはランダムであり、且つあらゆるチップで予測不能であるということである。したがって、各チップで後続のRをどのようにして生成するかが問題になってくる。

10

【1147】

基本的なケースでは、Rは全く変更されない。その結果としてR及び $F_{K1}[R]$ は、Random[]の各呼び出しに対して同じである。それらが同一である場合、 $F_{K1}[R]$ は、計算されるのではなく、定数にすることができる。攻撃者は、有効なルックアップテーブルを生成するために、単一の有効認証チップを使用し、そのシステムのために特にプログラミングされた複製チップ内でそのルックアップテーブルを使用することが可能である。定数Rは安全ではない。

【1148】

Rを変化させる最も簡単な概念的な方法は、Rを1ずつインクリメントすることである。Rは乱数から始められるので、種々のシステムにおけるその値はランダムである可能性が高い。しかし、初期Rを与えると、後続の全てのR値は直接的に決定できる（10000回繰り返すまでもなく、Rは R_0 から $R_0 + 10000$ までの値をとる）。Rをインクリメントすることは、定数Rに対する上記の攻撃を受ける心配はない。Rは常に異なるので、代用される複製チップと同数の認証チップを無駄にすること無しに、特定のシステムに対するルックアップテーブルを構築する方法がない。

20

【1149】

アダーを使用するインクリメントの他に、Rを変化させる別の方法は、それをLFSR（線形フィードバックシフトレジスタ）として実現することである。これは、アダーよりもシリコンが少ないという利点があり、また、LFSRの値ドメインは順次アクセスによって決定されるので、攻撃者が特定のシステムに対するRのレンジを直接的に決定できないという利点もある。所定の初期Rが生成する値を決定するためには、攻撃者は、可能性を反復し、それらを列挙しなければならない。Rを変化させることの利点は、LFSRソリューションの場合にも明白である。Rは常に相違するので、（特定のシステムに専用の）複製チップが置換する認証チップと同数の認証チップを使い果たすこと無しに、特定のシステムに対するルックアップテーブルを構築する方法がない。したがって、Rを変更するために、これ以上複雑な関数を導入する利点はない。関数とは無関係に、攻撃者は、常に、シミュレーションによって値の組を、耐用期間を通じて繰り返すことが可能である。本来の安全性は、Rの初期ランダム性にある。Rを変更するためLFSRを使用することは、（アダーよりもシリコン使用量が少ない点を別にして）連続した数値レンジに限定されないという利点があるだけである（即ち、R、 R_N を計算によって直接知ることができないので、攻撃者はLFSRをN回繰り返さなければならない）。

30

40

【1150】

したがって、認証チップ内の乱数生成器は、160ビットのLFSRである。最大周期LFSR用の160ビットのタップ選択（即ち、LFSRは、0が有効な状態ではないので、全部で $2^{160} - 1$ 状態を循環する）によって、図173に示めされるようなビット159、4、2及び1が得られる。LFSRは疎構造であり、多数のビットがフィードバックに使用されるわけではない（160ビットの中から4ビットだけが使用される）。これは、暗号アプリケーションの場合に問題であるが、非順次数生成のアプリケーションの場合には問題ではない。Rの160ビットシード値は0以外の任意の乱数を取り得る。なぜ

50

ならば、0が詰められたLFSRは0の制限のないストリームを生成するからである。上述のLFSRは最大周期LFSRであるため、全ての160ビットをそのままRとして使用することができる。出力ビット b_0 から順番に数を構築する必要がある。TSTの呼び出しが成功する毎に、乱数(R)はビット1、2、4及び159をXOR論理演算し、その結果を上位ビットへシフトすることによって進められる。新しいR及び対応した $F_{K_1}[R]$ は、次のRandomへの呼び出しの際に取り出すことができる。

【1151】

論理的攻撃への対抗

プロトコル3は認証チップによって使用される認証スキームである。このため、論理的手段による無効化に対し耐性であるべきである。プロトコル3に対する様々なタイプの攻撃の効果を説明したが、このセクションでは、プロトコル3の観点から攻撃の各タイプを詳述する。

10

【1152】

ブルートフォース攻撃

ブルートフォース攻撃はプロトコル3を破ることが保証されている。しかし、鍵の長さは、攻撃者がブルートフォース攻撃を実行するために要する時間が非常に長いので労力に値しない、ということの意味する。攻撃者は、複製認証チップを構築するため K_2 を破るだけでよい。 K_1 は、その他の形態の攻撃に対して K_2 を強化するために設けられているだけである。 K_2 へのブルートフォース攻撃は、したがって、160ビット鍵を破らなければならない。 K_2 に対する攻撃は、最大で 2^{160} 回の試行が必要であり、 2^{159} 回の試行後に鍵を検出する確率は50%である。1兆個のプロセッサのアレイを想定すると、各々が毎秒100万回のテストを実行すると、 2^{159} 回(7.3×10^{47})のテストには 2.3×10^{23} 年を要し、これは宇宙の寿命よりも長い。世界には1億台のパーソナルコンピュータしか存在しない。それらが攻撃の際に(例えば、インターネットを介して)接続されたとしても、その数は、上述の1兆個のプロセッサの攻撃よりも10000倍少ない。更に、1兆個のプロセッサの製造がナノコンピュータの時代に可能になったとしても、鍵を獲得するために要する時間は宇宙の寿命よりも長い。

20

【1153】

推測攻撃

攻撃者が単に「鍵を推測」することは理論的に可能である。實際上、十分な時間があり、あらゆる数を試すならば、攻撃者は鍵を獲得するであろう。これは、上述のブルートフォース攻撃と同じであり、50%の成功率を達成する前に、 2^{159} 回の試行をしなければならない。ある人が最初の試行で単に鍵を推測する機会みは 2^{160} である。これに対して、ある人が米国宝くじで一等賞を獲得し、同じ日に稲光によって死亡する確率は、 2^2 分の1に過ぎない。ある人が最初の試行で鍵を推測する確率は、 2^{160} 分の1であり、これは、地球上の全ての原子の中から二人が全く同一の原子を選択することに匹敵することであり、即ち、非常に可能性が低い。

30

【1154】

量子コンピュータ攻撃

K_2 を破るため、適切なアルゴリズムに組み込まれた160キュビットを含む量子コンピュータを構築することが必要である。160ビット鍵に対する攻撃は実施不可能である。量子コンピュータの可能性のぎりぎりの見込みは、50年以内に50キュビットを達成することである。50キュビット量子コンピュータを使用したとしても、160ビット鍵を解読するためには、 2^{110} 回のテストが必要になる。1億台の50キュビット量子コンピュータのアレイを想定し、各量子コンピュータが1マイクロ秒に 2^{50} 回の鍵(現時点での最も乱暴な推定を超えている)を試すことができるとするならば、鍵を見つけるために平均で180億年を要する。

40

【1155】

暗号文だけの攻撃

攻撃者は、RND及びRDへの呼び出しを呼び出し監視することによって K_1 に対する暗

50

号文だけの攻撃に出ること、及び R D 及び T S T への呼び出しを監視することによって K_2 に対する暗号文だけの攻撃に出ることが可能である。しかし、これらの全ての呼び出しが、平文と、平文のハッシュ形式を明らかにするならば、攻撃は、より強力な攻撃の形態、即ち、既知平文攻撃に変わるであろう。

【1156】

既知平文攻撃

ロジックアナライザを、システムと認証チップの間の配線に接続し、これにより、データのフローを監視することは容易である。このデータのフローは、既知平文と、平文のハッシュ形式とを生じ、 K_1 と K_2 の両方に対する既知平文攻撃に出るため使用できる。 K_1 に対する攻撃に出るため、R N D 及び T S T への多数の呼び出しを行うことが必要である（T S T への呼び出しが成功し、有効チップ上の R D への呼び出しが必要になる）。これは簡単であり、攻撃者は、システム認証チップと消耗品認証チップの両方をもつことが必要である。 K_1 、X、 $H_{K_1}[X]$ の各ペアが明らかになると、 K_2 、Y、 $H_{K_2}[X]$ のペアも明らかになる。攻撃者は更なる解析のためこれらのペアを収集しなければならない。このデータを用いて意味のある攻撃に出るためには何個のペアを収集すべきであるかという問題が生じる。統計的解析のためのデータの収集を必要とする攻撃の一例は差分暗号解読である。しかし、S H A - 1 又は H M A C - S H A 1 に対する攻撃は知られていないので、現時点では、収集されたデータの使い道はない。

【1157】

選択平文攻撃

暗号解読者が先行の調査の結果に基づいて後続の選択平文を変更する能力を備えているならば、 K_2 は、単純な選択平文攻撃よりも強い攻撃の形式である適応的選択平文攻撃の一部の形式に無防備である。選択平文攻撃は K_1 に対しては起こり得ない。なぜならば、呼び出し元には、R N D 関数への入力として使用される R を変更する方法がないからである（R N D は K_1 によるハッシュ化の結果を与える唯一の関数である）。R をクリアすることは、鍵をクリアする効果があるので有用ではなく、S S I コマンドは新しい R 値を格納する前に C L R を呼び出す。

【1158】

適応的選択平文攻撃

このタイプの攻撃は K_1 に対して起こり得ない。なぜならば、 K_1 は選択平文攻撃を許さないからである。しかし、この攻撃の一部の形式は K_2 に対して起こる可能性があり、特に、システムと消耗品の両方は、典型的に、攻撃者が入手可能であるからである（システムは、特定の車のようなある種の場合には攻撃者からアクセスできない）。H M A C 構造は、選択平文攻撃のあらゆる形式に対して安全性を提供する。その理由は主として、H M A C 構造が 2 個の秘密入力変数（元のハッシュの結果と秘密鍵）を有するからである。このようにして、入力変数が秘密であるときに、ハッシュ関数自体の衝突を検出することは、平文ハッシュ関数の衝突を見つけることよりも難しい。その理由は、前者は、S H A - 1 から入力／出力のペアを生成するため、S H A - 1 への直接アクセス（プロトコル 3 では許可されない）を必要とするからである。攻撃者によって収集できる唯一の値は、H M A C [R] 及び H M A C [R | M] である。これらは、S H A - 1 ハッシュ関数自体への攻撃ではなく、収集されたデータ間の統計的差分を調べる差分暗号解読攻撃まで攻撃を低減させる。S H A - 1 又は H M A C に対して知られている差分暗号解読攻撃がないとすれば、プロトコル 3 は適応的選択平文攻撃に対抗する。

【1159】

意図的なエラー攻撃

攻撃者は、T S T 関数及び R D 関数に対しては意図的なエラー攻撃しか出すことができない。なぜならば、これらは、鍵に対する入力を検証する唯一の関数であるからである。T S T と R D の両方の関数の場合、0 値は、入力にエラーが見つかったときに生成され、それ以上の情報は与えられない。その上、0 結果を生成するために要する時間は入力に依存せず、攻撃者はどのビットにエラーが存在するかに関する情報を得られない。したがって

、意図的なエラー攻撃は効果がない。

【1160】

連鎖攻撃

全ての形式の連鎖攻撃は、ハッシュされるべきメッセージが数ブロックに広がっているか、又は入力変数を何らかの方法でセットできることを前提としている。プロトコル3で使用されるHMAC-SHA1アルゴリズムは、同時に単一の512ビットのブロックしかハッシュできない。その結果として、連鎖攻撃はプロトコル3に対してはあり得ない。

【1161】

バースデー攻撃

HMACに対して知られている最強の攻撃は、ハッシュ関数の衝突の頻度に基づくバースデー攻撃である。しかし、これは、SHA-1のような必要最小限度で合理的なハッシュ関数には全く实际的でない。また、バースデー攻撃は、攻撃者が署名付きのメッセージを管理するときに限り実現可能である。プロトコル3は、デジタル署名の形式としてハッシングを使用する。システムは、有効人種チップからのレスポンスに組み込まれるべき数を送信する。認証チップはH[R|M]を用いて応答しなければならないが、入力値Rをコントロールできないので、バースデー攻撃は実現不能である。なぜならば、メッセージは、事実上、既に生成され、署名されているからである。その代わりに、攻撃者は、同じ値にハッシュした衝突メッセージを探索しなければならない（あなたと同じ誕生日の人を見つける場合と類似している）。したがって、複製チップは、 R_2 のハッシュと選択された M_2 がH[R|M]と同じハッシュ値を生ずるような新しい値 R_2 を見つけようとしなければならない。しかし、システム認証チップは正しいハッシュ値を明らかにしない（TS関数はハッシュ値が正しいかどうか依存して1又は0を返すだけである）。したがって、（衝突を見つけるために）正しいハッシュ値を見つける唯一の方法は、本物の認証チップに問い合わせることである。しかし、正しい値を見つけることは、Mを更新することを意味し、Mのデクリメントオンリー部分、及びMのリードオンリー部分を変更できないので、複製消耗品は、衝突を見つけようとする前に、本物の消耗品を更新しなければならない。代案は、成功を見つけるためのTS関数へのブルートフォース攻撃探索である（各複製消耗品はシステム消耗品へアクセスできることが必要である）。ブルートフォース検索は、上述の通り、この例では認証毎に、宇宙の寿命よりも長い時間を要する。ハッシュ値を適時に収集することは本物の消耗品をデクリメントしなければならないことを意味するので、複製消耗品がこの種の攻撃を仕掛ける意味がない。

【1162】

完全ルックアップテーブルによる代用

各システムの乱数シードは160ビットである。認証チップの最悪ケースの状況では、状態データが変更されない。その結果、一定値がMとして戻される。しかし、複製チップは、160ビット値である $F_{K_2}[R|M]$ を返し続けなければならない。結果が160ビットの160ビットルックアップテーブルを想定すると、これは、 7.3×10^{48} バイト、即ち、 6.6×10^{36} テラバイトを必要とし、近い将来に実施可能である空間よりも間違いなく大きい。これは、勿論、ROMのための値を収集する方法さえを考慮していない。したがって、完全なルックアップテーブルは完全に実現不能である。

【1163】

疎ルックアップテーブルによる代用

疎ルックアップテーブルは、認証チップへ送信されたメッセージが、実効的にランダムではなく、ある程度予測可能である場合に限り実施可能である。乱数Rは、自然界のランダム事象から収集された未知乱数でシードされる。複製品製造者は全てのしすてむに対して考えられるRのレンジを知る可能性はない。なぜならば、各ビットが1であるか、又は0である確率は50%であるからである。全てのシステムにおけるRのレンジは未知であるため、全てのシステムで使用可能な疎ルックアップテーブルを構築することは不可能である。したがって、一般的な疎ルックアップテーブルによる攻撃は考えられない。しかし、複製品製造者は、ある特定のシステムに対するRのレンジを知ることは可能である。これ

は、L F S Rに、特定のシステム認証チップのR N D関数への呼び出しからの現在の結果をロードし、将来に向けて数回繰り返すことによって実現できる。これが行われた場合、特定のRのレンジに対するレスポンスだけを格納した特別なR O M、即ち、その特定のシステムの消耗品に専用のR O Mを構築できる。しかし、攻撃者は、正しい情報をR O Mに収容し続けなければならない。したがって、攻撃者は、有効認証チップを検出し、Rの値毎にそれを呼び出すことが必要である。

【1164】

複製認証チップが完全な消耗品をレポートし、接続のロスと新しい完全な消耗品の挿入をシミュレーションする前に1回だけ使用できる場合を想定する。複製消耗品は、完全な消耗品の認証に対するレスポンス、及び部分的に使用された消耗費にに対するレスポンスを格納しなければならない。最悪ケースで、R O Mは、システムの耐用期間に亘るRに関して、完全な消耗品及び部分的に使用された消耗品のエントリーを格納する。しかし、有効認証チップは、情報を生成するために使用され、部分的にそのプロセスで使用されるべきである。所定のシステムがn個のR値だけを生成するならば、必要な疎ルックアップR O Mは、Mに対する異なる値の個数によって乗算された10nバイトである。R O Mを構築するために要する時間は、R D呼び出しの間に課された時間に依存する。

【1165】

それにもかかわらず、複製品製造者は、詰め替えのため返品される消耗品に頼らざるを得ない。なぜならば、最初の場所にR O Mを構築するコストは1個の消耗品を消費するからである。このような状況における複製品製造者のビジネスは、結果的に詰め替えにある。時間及びコストは、Rのサイズと、ルックアップに組み込まれるべきMの値の種類の数と、に依存する。その上、カスタム複製R O Mは、一つ一つのシステムに適合する必要がある、（完全なデータ及び部分的に使用されたデータを提供するため）異なる有効認証チップがシステム毎に使用されなければならない。システム内の認証チップの使用は、したがって、この種の攻撃が複製品製造者にとって価値があるかどうかを判定するために検査しなければならない。一例は、耐用期間に約10000プリントを有するカメラシステムである。単一のデクリメントオンリー値（残りプリント数）と、R Dの呼び出しの間に1秒の遅延がある場合を想定する。このようなシステムにおいて、疎テーブルは、構築に約3時間を要し、100Kを消費する。R O Mを構成するためには、有効認証チップを消費することが必要であるので、料金は、1個の消耗品と複製消耗品の合計よりも高くしなければならないことに注意すべきである。このようにして、1個の消耗品に対してこの機能を実行することは（複製消耗品が同等の多数の認証消耗品に何とかして収容されない限り）費用対効果がよくない。複製品製造者がシステムの所有者毎にカスタムR O Mを構築する労力を惜しまないのであれば、より簡単なアプローチは、認証チップを完全に無視するようにシステムを更新することであろう、したがって、この攻撃は、システム毎の攻撃として実現可能であり、所定のシステム／消耗品の組み合わせに対してこのようなことが起こる可能性を判定しなければならない。その可能性は、消耗品及び認証チップのコストと、消耗品の寿命と、消耗品の利益幅と、R O Mを生成するために要する時間と、得られるR O Mのサイズと、顧客が同じ複製チップ等を使用する詰め替えのために複製品製造者へ戻ってくるかどうかと、に依存する。

【1166】

差分暗号解読

既知差分攻撃は、D E S及びその他の同様のアルゴリズムで使用されるようなSボックスの構造にかなり依存している。プロトコル3で使用されるH M A C - S H A 1のような他のアルゴリズムはSボックスを持たないが、攻撃者は、

最小差分入力及びそれに対応した出力

最小差分出力及びそれに対応した入力

の統計的解析を行うことによって差分的な攻撃を実行し得る。

【1167】

この性質の攻撃に出るため、入力／出力ペアの組が収集される。プロトコル3からの収集

10

20

30

40

50

は、既知平文攻撃、又は部分適応的選択平文攻撃による。明らかに、後者が選ばれる方が有用である。一般的にハッシュ化アルゴリズムは差分解析に対抗するように設計されている。特に、SHA-1は、特に80語の拡張によって、強化されているので、入力における最小限の差は、(128ビットハッシュ関数と比較して)多数のビット位置が変わる出力を生成するであろう。その上、収集される情報は、HMACアルゴリズムの性質により、直接的なSHA-1入力/出力ではない。HMACアルゴリズムは、既知の値を、未知の値(鍵)でハッシュし、このハッシュの結果が次に別個の未知の値で再ハッシュされる。攻撃者は、秘密値も、最初のハッシュの結果もわからないので、SHA-1からの入力及び出力は不明であり、あらゆる差分攻撃を極めて難しくさせる。以下では、認証チップからの最小限度の差の入力及び出力をより詳細に説明する。

10

【1168】

最小限度の差の入力

この場合、攻撃者は、 X 値、 $F_K[X]$ 値の組を取得し、 X 値が最小限度の差であり、出力 $F_K[X]$ の間の統計的な差を検査する。攻撃は、最小限度のビット数だけの差がある X 値に依存する。 $F_K[X]$ 値を比較するために最小限度の差の X 値をどのようにして獲得するかが問題である。

【1169】

K_1 : K_1 の場合、攻撃者は、最小限度の差 X 、 $F_{K_1}[X]$ ペアを統計的に検査しなければならない。しかし、攻撃者は、 X 値を選択できず、関連した $F_{K_1}[X]$ 値を獲得できない。 X 、 $F_{K_1}[X]$ ペアは、システム認証チップ上の RND 関数を呼び出すことによってのみ生成することができるので、攻撃者は RND を複数回呼び出し、観測された各ペアをテーブルに記録する。次に、 $F_{K_1}[X]$ の統計的解析を行うため、観測された値の中で十分に最小限度の差 X 値を探索しなければならない。

20

【1170】

K_2 : K_2 の場合、攻撃者は、最小限度の差 X 、 $F_{K_2}[X]$ ペアを統計的に検査しなければならない。 X 、 $F_{K_2}[X]$ ペアを生成する唯一の方法は、RD 関数によるものであり、所定の Y 、 $F_{K_1}[Y]$ ペアに対して $F_{K_2}[X]$ を生成し、ここで、 $X = Y \mid M$ である。これは、 Y と、 M の変更可能部分が攻撃者によって限られた範囲で選択され得ることを意味する。選択の量は、したがって、できる限り多くされるべきである。

【1171】

攻撃者の選択を制限する第1の方法は Y を制限することである。なぜならば、RD はフォーマット Y 、 $F_{K_1}[Y]$ の入力を要求するからである。有効なペアは RND 関数から容易に獲得できるが、それは、RND の選択によるペアである。攻撃者は、RND から適切なペアを獲得した場合に限り、又は K_1 を知っている場合に限り、固有の Y を提供することが可能である。RND から適切なペアを獲得するためには、ブルースフォース探索が必要になる。 K_1 を知ることは、RND 関数から獲得されたペアへの暗号解読を実行することによってのみ、即ち、事実上の既知テキスト攻撃によってのみ論理的に実現可能である。RND は1秒当たり呼び出せる回数は決まっているが、 K_1 はシステムのチップで共通である。したがって、既知ペアは並列に生成できる。攻撃者の選択を制限する第2の方法は、 M を制限すること、又は少なくとも攻撃者の M 選択能力を制限することである。 M の制限は、 M の一部をリードオンリーにさせ、認証チップ毎に相違させ、 M の他の部分をデクリメントオンリーにすることで実現される。 M のリードオンリー部分は、理想的には、認証チップ毎に異なるべきであるので、シリアル番号、バッチ番号、又は乱数のような情報でもよい。 M のデクリメントオンリー部分は、攻撃者が異なる M を試すとき、その回数だけ M のその部分だけをデクリメントすることができ、 M のデクリメントオンリー部分が0まで低下した後、その部分は再度変更し得ない、ということ意味する。新しい認証チップ53を獲得すると、新しい M を与えるが、リードオンリー部分は、前の認証チップのリードオンリー部分とは異なり、したがって、攻撃者の M を選択する能力が更に低下する。その結果として、攻撃者は、 Y 及び M に対する値を選択するチャンスが少ししか高めることができない。

30

40

50

【1172】

最小限度の差の出力

この場合、攻撃者は、 X 値、 $F_K[X]$ 値の組を取得し、 $F_K[X]$ 値が最小限度の差を表し、 X 値の間の統計的な差を検査する。攻撃は、最小限度のビット数だけの差がある $F_K[X]$ 値に依存する。 K_1 と K_2 の両方に対し、攻撃者は所与の $F_K[X]$ に対する X 値を生成する方法がない。そうするためには、 F は一方向関数であるという事実には違反するであろう。その結果として、攻撃者がこの性質をもつ攻撃を組み込む唯一の方法は、全ての観測された X 、 $F_K[X]$ ペアをテーブルに記録することである。検索は、 X 値の統計的解析を行うため十分に最小限度の差 $F_K[X]$ 値を見つけるため観測値の中で行われる。これが (M に関する制限及び R の選択に関する制限のために非常に限定されている) 最小限度の差の入力攻撃よりも多くの作業を必要とするならば、この攻撃は無益である。

10

【1173】

メッセージ置換攻撃

この種の攻撃を実行するためには、複製物製造者は本物の認証チップ 53 を格納する必要があるが、それは減少しないので、事実上、再使用可能である。複製認証チップは、メッセージを盗み取り、それを自分のメッセージで置き換える。しかし、この攻撃は攻撃者にとって成功しない。複製認証チップは、 WR コマンドを本物の認証チップへ渡さないように選択するかもしれない。しかし、後続の RD コマンドは (WR が成功したかのよう) 正しいレスポンスを戻さなければならない。正しいレスポンスを返すためには、特定の R 及び M についてのハッシュ値がわかっているなければならない。パースデー攻撃のセクションで説明したように、攻撃者は、本物のチップ内の M を実際に更新しなければハッシュ値を決定できないが、攻撃者はこれを行うことを望まない。システムによって送信された R を変更することさえ役に立たない。なぜならば、システム認証チップは引き続く TST の間に R をマッチさせなければならないからである。したがって、メッセージ置換攻撃は成功しない。これは、システムが消耗品を使用する前に消耗品残量を更新する場合に限り成り立つ。

20

【1174】

鍵生成器のリバースエンジニアリング

疑似乱数生成器が鍵を生成するため使用される場合、複製品製造者は、生成器プログラムを取得するか、又は使用されたランダムシードを推測する可能性がある。これは、ネットスケープのセキュリティプログラムが最初に破られたときの方法である。

30

【1175】

認証全体の回避

プロトコル 3 は、システムが消耗品の使用前に消耗品状態データを更新し、(書き込みを認証するため) 全ての書き込みの後に読み出しを続けることを要求する。このように、消耗品の使用毎に認証を要求する。システムがこれらの二つの簡単なルールに固執するならば、複製品製造者は (疎 ROM ルックアップのような) 上述の方法により認証をシミュレーションしなければならない。

【1176】

認証チップの再使用

上述のように、プロトコル 3 は、システムが消耗品の使用前に消耗品状態データを更新し、(書き込みを認証するため) 全ての書き込みの後に読み出しを続けることを要求する。このように、消耗品の使用毎に認証を要求する。消耗品が使い切られた場合、その認証チップの適切な状態データ値は 0 まで減少している。したがって、チップは別の消耗品で使用できない。このことが成り立つのは、デクリメントオンリーデータ項目を保持する認証チップの場合だけであることに注意する必要がある。使用される毎にデクリメントさせられる状態データが無い場合、チップの再使用を阻止するものはない。これは、プレゼンスオンリー認証と消耗品ライフタイム認証との間の基本的な差異である。プロトコル 3 はその両方を可能にさせる。結論として、消耗品がシステムによって使用されるデクリメントオンリーデータ項目を備えている場合、認証チップは、秘密鍵を知っている有効なプログ

40

50

ラミングステーションによって完全に再プログラミングされない限り、再使用不能である。

【1177】

コスト節約のため認証を省略する経営上の判断

外部攻撃には限られないが、コストを節約するため将来のシステムでは認証を省略する判断は、様々なマーケットに種々の影響を与えるであろう。大量生産消耗品の場合、販売開始後に認証を導入することは極めて困難であるということに留意しなければならない。なぜならば、認証された消耗品を要求するシステムは、流通中の旧式の消耗品と連携しないからである。同様に、ある段階で認証を打ち切ることも非現実的である。なぜならば、旧式のシステムは、新しい認証されていない消耗品と連携しないからである。第2のケースでは、旧式のシステムは、システム認証チップを、同じプログラミングインタフェースを具備しているが、そのTST関数が常に成功する単純なチップで置換することによって、個別に改造することが可能である。勿論、そのシステムは、常に成功するTST関数をテストし、停止するようにプログラミングしてもよい。車／車キー、又は扉／扉の鍵、又はその他の類似した状況のような特殊化したペアの場合、将来のシステムで認証を省くことは重要ではなく影響もない。なぜならば、消費者は、システムと消耗品認証チップの組全体を同時に購入しているからである。

10

【1178】

強盗／賄賂攻撃

この形態の攻撃は、以下の二つの環境の何れかでしか成功しない：

20

K_1 、 K_2 、及び R はチッププログラマによって既に記録されているか、又は攻撃者は、将来の K_1 、 K_2 、及び R の値を強制的に記録させることができる。

【1179】

プログラミングステーションの外部の人間又はコンピュータシステムが鍵を知らないとき、暴力や賄賂をもってしてもそれらを明らかにできない。この種の攻撃に対する安全性のレベルは、究極的に、システム／消耗品所有者が望ましいサービスのレベルに応じて行う判断である。例えば、自動車会社は、製造した全てのキーの記録を保存しようとするので、その車のため新しいキーの作成を要求することができる。しかし、これは、キーデータベースの全体が危険にさらされる可能性があり、攻撃者は製造者の既存の車に対するキーを作成できるようになる。これは、攻撃者が新しい車キーを作成することを可能にさせない。勿論、キーデータベース自体は、別の鍵で暗号化され、キーデータベースにアクセスするためには、一定の人数の人のそれぞれの鍵部分を一つに組み合わせることが必要である。特定の車にどのキーが使用されているかについての記録が保存されていないならば、キーを紛失した場合に補助キーを作成する方法はない。この場合、所有者は自分の車の認証チップと自分の車キーの全てを交換しなければならないであろう。これは、必ずしも困った状況ではない。これに対して、プリンタのインクカートリッジのような消耗品の場合、一つの鍵の組み合わせが全てのシステム及び全ての消耗品に対して使用される。勿論、鍵のバックアップが保存されていないならば、誰もその鍵がわからず、攻撃はあり得ない。しかし、バックアップの無い状況は、インクカートリッジのような消耗品の場合に望ましくない。なぜならば、鍵が失われた場合、それ以上消耗品を製造できないからである。したがって、製造者は鍵情報のバックアップを幾つかの部分に保存すべきであり、何人かの人は、鍵全体の情報を明らかにするため、それぞれの部分を一つに結合することが必要である。これは、チッププログラミングステーションに再ロードしなければならない場合に要求される。いずれにしても、これらの攻撃はプロトコル3自体に対する攻撃ではない。なぜならば、人が認証プロセスに関与していないからである。そのかわりに、これは、チップのプログラミングステージに対する攻撃である。

30

40

【1180】

HMAC-SHA1

認証のためのメカニズムは、HMAC-SHA1アルゴリズムであり、

HMAC-SHA1(R_1 , K_1)又は

50

HMAC-SHA1 (R | M, K₂)

の一方に従って動作する。

【1181】

次に、HMAC-SHA1アルゴリズムをここまでの説明よりも詳しく検討し、本来の定義よりもメモリリソース必要量が少なくなるアルゴリズムの最適化を説明する。

【1182】

HMAC

HMACアルゴリズムは、以下の定義を前提として進む：

H = ハッシュ関数（例えば、MD5又はSHA-1）；

n = Hから出力されるビット数（例えば、SHA-1の場合に160ビット、MD5の場合に128ビット）； 10

M = MAC関数が適用されるデータ；

K = 両当事者で共有される秘密鍵；

ipad = 0x36の64回の繰り返し；

opad = 0x5Cの64回の繰り返し。

【1183】

HMACアルゴリズムは次の通りである：

0x00バイトをKの最後に追加することによりKを64バイトに拡張する；

(1) で作成された64バイト文字列をipadでXOR論理演算する；

(2) で作成された64バイト文字列にデータストリームMを追加する； 20

(3) で生成されたストリームにHを適用する；

(1) で作成された64バイト文字列をopadでXOR論理演算する；

(4) からの結果Hを(5)から得られた64バイト文字列に追加する；

(6) の出力にHを追加し、その結果を出力する。

【1184】

これにより、

$HMAC[M] = H[(KAopad) | H[(KAipad) | M]]$

である；

HMAC-SHA1アルゴリズムは、単にH = SHA-1となるHMACである。

【1185】 30

SHA-1

SHA-1ハッシングアルゴリズムは以下で要約するアルゴリズムに規定される。

【1186】

9個の32ビット定数が定義される。5個の定数は連鎖変数を初期化するため使用され、4個の付加的定数がある。

【1187】

【表104】

	初期連鎖値		付加的定数
h_1	0x67452301	y_1	0x5A827999
h_2	0xEFCDAB89	y_2	0x6ED9EBA1
h_3	0x98BADCFE	y_3	0x8F1BBCDC
h_4	0x10325476	y_4	0xCA62C1D6
h_5	0xC3D2E1F0		

40

最適化されていないSHA-1は、全部で2912ビットのデータ記憶を必要とする：

5個の32ビット連鎖変数： H_1 、 H_2 、 H_3 、 H_4 及び H_5 が定義される；

5個の32ビット作業変数：A、B、C、D及びEが定義される；

1個の32ビット一時変数：tが定義される；

80個の32ビット一時レジスタ： $X_0 - 79$ が定義される。 50

【1188】

以下の関数はSHA-1のために定義される。

【1189】

【表105】

記号名	説明
+	加算モジュロ 2^{32}
$X \cup Y$	Yビット位置分のX左回転の結果
$f(X, Y, Z)$	$(X \wedge Y) \vee (\sim X \wedge Z)$
$g(X, Y, Z)$	$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$
$h(X, Y, Z)$	$X \oplus Y \oplus Z$

10

ハッシングアルゴリズムは、最初に、512ビットの倍数になるまで入力メッセージをパディングし、連鎖変数 $H_1 - 5$ を $h_1 - 5$ で初期化する。パッドされたメッセージは、次に、512ビットチャンクで処理され、出力ハッシュ値は、連鎖変数： $H_1 \parallel H_2 \parallel H_3 \parallel H_4 \parallel H_5$ の連結によって与えられる最後の160ビット値である。次に、SHA-1アルゴリズムのステップを詳細に説明する。

【1190】

ステップ1 処理

SHA-1の第1ステップは、以下のように入力メッセージを512ビットの倍数になるまで詰め込み、連鎖変数を初期化する。

20

【1191】

【表106】

入力メッセージを処理するため行うステップ	
入力メッセージ埋め込み	メッセージに1ビットを付加
	埋め込みメッセージの長さが512ビットの倍数の64ビットショートとなるように0ビットを付加
	元の入力メッセージのビットの長さを含む64ビット値を付加。最上位ビットから最下位ビットとして長さを格納。
連鎖変数初期化	$H_1 \leftarrow h_1, H_2 \leftarrow h_2, H_3 \leftarrow h_3, H_4 \leftarrow h_4, H_5 \leftarrow h_5$

30

ステップ2 処理

詰め込まれた入力メッセージが処理可能である。512ビットのブロック内でメッセージを処理する。各512ビットのブロックは、 16×32 ビット語の形であり、 $Input\ Word_0 - 15$ として表される。

【1192】

【表 1 0 7】

512 ビットブロック (InputWord ₀₋₁₅) 毎に行うステップ	
512 入力ビットを X ₀₋₁₅ へコピー	For j = 0 to 15 X _j = InputWord _j
X ₀₋₁₅ を X ₁₆₋₇₉ へ拡張	For j = 16 to 79 X _j ← ((X _{j-3} ⊕ X _{j-8} ⊕ X _{j-14} ⊕ X _{j-16}) ∪ 1)
作業変数初期化	A ← H ₁ , B ← H ₂ , C ← H ₃ , D ← H ₄ , E ← H ₅
第 1 ラウンド	For j = 0 to 19 t ← ((A ∪ 5) + f(B, C, D) + E + X _j + y ₁) E ← D, D ← C, C ← (B ∪ 30), B ← A, A ← t
第 2 ラウンド	For j = 20 to 39 t ← ((A ∪ 5) + h(B, C, D) + E + X _j + y ₂) E ← D, D ← C, C ← (B ∪ 30), B ← A, A ← t
第 3 ラウンド	For j = 40 to 59 t ← ((A ∪ 5) + g(B, C, D) + E + X _j + y ₃) E ← D, D ← C, C ← (B ∪ 30), B ← A, A ← t
第 4 ラウンド	For j = 60 to 79 t ← ((A ∪ 5) + h(B, C, D) + E + X _j + y ₄) E ← D, D ← C, C ← (B ∪ 30), B ← A, A ← t
連鎖変数更新	H ₁ ← H ₁ + A, H ₂ ← H ₂ + B H ₃ ← H ₃ + C, H ₄ ← H ₄ + D H ₅ ← H ₅ + E

10

20

ステップ 3 終了

詰め込まれた入力メッセージの全ての 512 ビットのブロックが処理された後、出力ハッシュ値は、H₁ | H₂ | H₃ | H₄ | H₅ によって与えられる最後の 160 ビット値である。

【1193】

30

ハードウェア実装用の最適化

SHA-1 のステップ 2 の手続は、ハードウェアのために最適化されていない。特に、80 個の一時的な 32 ビットレジスタは、ハードウェア実装上の貴重なシリコンを使い果たす。このセクションは、16 個の一時レジスタだけを使用する SHA-1 アルゴリズムへの最適化を説明する。シリコンの削減は、2560 ビットから 512 ビットまでへの減少であり、2000 ビット以上の節約になる。これは、一部のアプリケーションでは重要でないが、認証チップの場合、記憶スペースはできるだけ縮小されるべきである。最適化の基礎は、元の 16 ワードメッセージブロックが 80 ワードメッセージブロックに拡張されるが、80 ワードはアルゴリズムの間に更新されないことにある。その上、ワードは、前の 16 ワードだけに依存しているので、拡張されたワードは、16 ワードが後方参照のため保持される限り、処理中にオンザフライ方式で計算可能である。どのレジスタを使おうとしているかを追跡するために循環カウンタが必要になるが、その効果は大量の記憶量を節約することである。単一の値 j で X をインデックス付けするのではなく、繰り返しを通してカウントするため 5 ビットのカウンタを使用する。これは、5 ビットレジスタを 16 又は 20 で初期化し、0 に達するまでそれをデクリメントすることによって実現される。16 個の一時変数を 80 個であるかのようにして更新するため、4 個のインデックスが必要であり、各インデックスは 4 ビットレジスタである。4 個の全てのインデックスはアルゴリズムの途中（ラップアラウンド付きで）インクリメントされる。

40

【1194】

【表 108】

512 ビットブロック (InputWord ₀₋₁₅) 毎に行うステップ	
作業変数初期化	$A \leftarrow H_1, B \leftarrow H_2, C \leftarrow H_3, D \leftarrow H_4, E \leftarrow H_5$ $N_1 \leftarrow 13, N_2 \leftarrow 8, N_3 \leftarrow 2, N_4 \leftarrow 0$
第 0 ラウンド 512 入力ビットを X_{0-15} へコピー	16 回実行 $X_{N4} = \text{InputWord}_{N4}$ $[\hat{N}_1, \hat{N}_2, \hat{N}_3]_{\text{optional}} \hat{N}_4$
第 1A ラウンド	16 回実行 $t \leftarrow ((A \cup 5) + f(B, C, D) + E + X_{N4} + y_1)$ $[\hat{N}_1, \hat{N}_2, \hat{N}_3]_{\text{optional}} \hat{N}_4$ $E \leftarrow D, D \leftarrow C, C \leftarrow (B \cup 30), B \leftarrow A, A \leftarrow t$
第 1B ラウンド	4 回実行 $X_{N4} \leftarrow ((X_{N1} \oplus X_{N2} \oplus X_{N3} \oplus X_{N4}) \cup 1)$ $t \leftarrow ((A \cup 5) + f(B, C, D) + E + X_{N4} + y_1)$ $\hat{N}_1, \hat{N}_2, \hat{N}_3, \hat{N}_4$ $E \leftarrow D, D \leftarrow C, C \leftarrow (B \cup 30), B \leftarrow A, A \leftarrow t$
第 2 ラウンド	20 回実行 $X_{N4} \leftarrow ((X_{N1} \oplus X_{N2} \oplus X_{N3} \oplus X_{N4}) \cup 1)$ $t \leftarrow ((A \cup 5) + f(B, C, D) + E + X_{N4} + y_2)$ $\hat{N}_1, \hat{N}_2, \hat{N}_3, \hat{N}_4$ $E \leftarrow D, D \leftarrow C, C \leftarrow (B \cup 30), B \leftarrow A, A \leftarrow t$
第 3 ラウンド	20 回実行 $X_{N4} \leftarrow ((X_{N1} \oplus X_{N2} \oplus X_{N3} \oplus X_{N4}) \cup 1)$ $t \leftarrow ((A \cup 5) + g(B, C, D) + E + X_{N4} + y_3)$ $\hat{N}_1, \hat{N}_2, \hat{N}_3, \hat{N}_4$ $E \leftarrow D, D \leftarrow C, C \leftarrow (B \cup 30), B \leftarrow A, A \leftarrow t$
第 4 ラウンド	20 回実行 $X_{N4} \leftarrow ((X_{N1} \oplus X_{N2} \oplus X_{N3} \oplus X_{N4}) \cup 1)$ $t \leftarrow ((A \cup 5) + h(B, C, D) + E + X_{N4} + y_4)$ $\hat{N}_1, \hat{N}_2, \hat{N}_3, \hat{N}_4$ $E \leftarrow D, D \leftarrow C, C \leftarrow (B \cup 30), B \leftarrow A, A \leftarrow t$
連鎖変数更新	$H_1 \leftarrow H_1 + A, H_2 \leftarrow H_2 + B$ $H_3 \leftarrow H_3 + C, H_4 \leftarrow H_4 + D$ $H_5 \leftarrow H_5 + E$

10

20

30

ラウンド 0 及び 1 A の間の N_1 、 N_2 及び N_3 のインクリメントは随意的である。ソフトウェア実装は、時間を要するので、それらをインクリメントせずに、16 巡のループの最後に、全 4 個のカウンタは元の値になる。ハードウェアの設計者は、制御ロジックを節約するため、全 4 個のカウンタを同時にインクリメントしようとする。呼び出し元が 512 ビットの X_{0-15} をロードするならば、ラウンド 0 は完全に省くことができる。

40

【1195】

HMAC-SHA1

認証チップ実装の場合、HMAC-SHA1 ユニットは、二つのタイプの入力： K_1 を使用する R、及び K_2 を使用する R|M にハッシングを実行するだけである。入力は 2 個の一定長さであるため、HMAC と SHA-1 をチップ上に別個のエンティティとして設けるのではなく、それらを合成し、ハードウェアを最適化することができる。SHA-1 のステップ 1 におけるメッセージのパディング（1 ビット、0 ビットの文字列、及びメッセージの長さ）は、異なるメッセージがパディング後に同じように見えないことを保証する

50

ために必要である。2タイプのメッセージしか取り扱わないので、パディングは定数0でもよい。その上、SHA-1アルゴリズムの最適化バージョンが使用され、16個の32ビットワードだけが一時記憶のために使用される。これらの16個のレジスタは、最適化されたHMAC-SHA1ハードウェアによって直接ロードされる。9個の32ビット定数 h_{1-5} 及び y_{1-4} は依然として必要であるが、それらが定数であることは、ハードウェア実装に有利である。ハードウェアの最適化されたHMAC-SHA-1は、全部で、以下の1024ビットのデータ記憶を必要とする：

5個の32ビット連鎖変数： H_1 、 H_2 、 H_3 、 H_4 及び H_5 が定義される；

5個の32ビット作業変数： A 、 B 、 C 、 D 及び E が定義される；

一時記憶及び最終結果のための5個の32ビット変数： $Buff160_{1-5}$ が定義される；

1個の32ビット一時変数： t が定義される；

16個の32ビット一時レジスタ： X_{0-15} が定義される。

【1196】

以下の2セクションは、HMAC-SHA1への2タイプの呼び出しのためのステップを説明する。

【1197】

$H[R, K_1]$

K_1 を使用してRの鍵付きハッシュを生成するケースでは、元の入力メッセージRは、160ビットの一定長さである。したがって、処理中にこのことを有利に活用することができる。SHA-1アルゴリズムの最初の部分の間に X_{0-15} をロードするのではなく、 X_{0-15} を直接ロードし、これにより、SHA-1の最適化プロセスブロック（ステップ2）のラウンド0を省く。擬似コードは次のステップを行う。

【1198】

【表109】

ステップ	説明	動作
1	Process $K \oplus \text{ipad}$	$X_{0-4} \leftarrow K_1 \oplus 0x363636...$
2		$X_{5-15} \leftarrow 0x363636...$
3		$H_{1-5} \leftarrow h_{1-5}$
4		Process Block
5	Process R	$X_{0-4} \leftarrow R$
6		$X_{5-15} \leftarrow 0$
7		Process Block
8		$Buff160_{1-5} \leftarrow H_{1-5}$
9	Process $K \oplus \text{opad}$	$X_{0-4} \leftarrow K_1 \oplus 0x5C5C5C...$
10		$X_{5-15} \leftarrow 0x5C5C5C...$
11		$H_{1-5} \leftarrow h_{1-5}$
12		Process Block
13	Process previous $H[x]$	$X_{0-4} \leftarrow \text{Result}$
14		$X_{5-15} \leftarrow 0$
15		Process Block
16	結果獲得	$Buff160_{1-5} \leftarrow H_{1-5}$

$H[R \parallel M, K_2]$

K_2 を使用して $R \parallel M$ の鍵付きハッシュを生成するケースでは、元の入力メッセージは、

416 (256 + 160) ビットの一定長さである。したがって、処理中にこのことを有利に活用することができる。SHA-1 アルゴリズムの最初の部分の間に X_{0-15} をロードするのではなく、 X_{0-15} を直接ロードし、これにより、SHA-1 の最適化プロセスブロック (ステップ 2) のラウンド 0 を省く。擬似コードは次のステップを行う。

【1199】

【表 110】

ステップ	説明	動作
1	Process $K \oplus \text{ipad}$	$X_{0-4} \leftarrow K_2 \oplus 0x363636...$
2		$X_{5-15} \leftarrow 0x363636...$
3		$H_{1-5} \leftarrow h_{1-5}$
4		Process Block
5	Process $R \parallel M$	$X_{0-4} \leftarrow R$
6		$X_{5-12} \leftarrow M$
7		$X_{13-15} \leftarrow 0$
8		Process Block
9		$\text{Temp} \leftarrow H_{1-5}$
10	Process $K \oplus \text{opad}$	$X_{0-4} \leftarrow K_2 \oplus 0x5C5C5C...$
11		$X_{5-15} \leftarrow 0x5C5C5C...$
12		$H_{1-5} \leftarrow h_{1-5}$
13		Process Block
14	Process previous $H[x]$	$X_{0-4} \leftarrow \text{Result}$
15		$X_{5-15} \leftarrow 0$
16		Process Block
17	結果獲得	$\text{Result} \leftarrow H_{1-5}$

データ記憶完全性

確認チップは、認証プロトコル 3 によって要求される変数を保持するため、不揮発性メモリを含む。以下の不揮発変数が定義される。

【1200】

10

20

30

【表 1 1 1】

変数名	サイズ (ビット単位)	説 明
M [0...15]	256	シリアル番号、メディア残量等のような状態データを含む 16 ワード (各々 16 ビット)
K ₁	160	認証中に R を変換するため使用される鍵
K ₂	160	認証中に M を変換するため使用される鍵
R	160	現在の乱数
Access Mode [0...15]	32	M [n]に対する 16 組の 2 ビット AccessMode 値
MinTicks	32	鍵ベース関数への呼び出しの間のクロックチックの最小数
SIWritten	1	セットされている場合、秘密鍵情報 (K ₁ , K ₂ 及び R) はチップに書き込まれている。クリアされている場合、秘密情報は未だ書き込まれていない。
IsTrusted	1	セットされている場合、RND 関数及び TST 関数を呼び出せるが、RD 関数及び WR 関数を呼び出せない。クリアされている場合、RND 関数及び TST 関数を呼び出せないが、RD 関数及び WR 関数を呼び出せる。
合計ビット数	802	

10

20

これらの変数がフラッシュメモリにあるならば、古い値を置き換えるため新しい値を書き込むことは簡単ではないことに注意する必要がある。メモリは、最初に消去され、次に、適切なビットをセットされる。これは、フラッシュメモリベースの変数を変更するため使用されるアルゴリズムに影響を与える。例えば、フラッシュメモリはシフトレジスタとして簡単に使用できない。汎用演算でフラッシュメモリ変数を更新するためには、以下のステップが必要になる：

N ビット値全体を汎用レジスタに読み込む；

汎用レジスタに演算を実行する；

変数に対応したフラッシュメモリを消去する；

汎用レジスタにセットされたビットに基づいてフラッシュメモリロケーションのビットをセットする。

30

【1 2 0 1】

認証チップのリセットはこれらの不揮発性変数に影響を与えない。

【1 2 0 2】

M とアクセスモード

変数 M [0] から M [15] は、シリアル番号、バッチ番号、及び消耗品残量のような消耗品状態データを保持するため使用される。各 M [n] レジスタは 16 ビットであり、M ベクトル全体で 256 ビット (32 バイト) になる。クライアントは個々の M [n] 変数を読み書きできない。その代わりに、M で表されるベクトル全体が単一論理アクセスによって読み出し、又は書き込みをされる。M は RD (リード) コマンドを使用して読み出され、WR (ライト) コマンドを使用して書き込まれる。これらのコマンドは、K₁ と K₂ の両方が確定し (SIWritten = 1)、認証チップが消耗品の信頼できないチップ (IsTrusted = 0) である場合に限り成功する。M は多数の異なるデータタイプを格納するが、それらは書き込み許可が異なるだけである。各データタイプは常に読み出すことができる。一旦クライアントメモリに入ると、256 ビットはクライアントによって選択された何らかの方法で解釈できる。M の 256 ビット全体は、認証の章で説明したようにセキュリティの理由から少量ずつ読み出すのではなく、同時に読み出される。様々な書き込み許可が以下の表に列挙されている。

40

【1 2 0 3】

【表 1 1 2】

データタイプ	アクセス注記
リードオンリー	絶対書き込み不可
リードライト	常に書き込み可
デクリメントオンリー	新しい値が古い値よりも小さい場合に限り書き込み可。デクリメントオンリー値は、典型的に 16 ビット又は 32 ビット値であるが、16 ビットの倍数でもよい。

書き込みに要求される保護を実現するため、2 ビットアクセスモード値が $M[n]$ 毎に定められる。以下の表は、2 ビットアクセスモードビットパターンの解釈を定義する。

【1 2 0 4】

【表 1 1 3】

ビット	Op	説明	書き込みコマンド中に行われる動作
00	RW	リードライト	新しい 16 ビット値は常に $M[n]$ に書き込まれる。
01	MSR	デクリメントオンリー(最上位領域)	新しい 16 ビット値は、現在の $M[n]$ の値よりも小さい場合に限り $M[n]$ に書き込まれる。これは、デクリメントオンリー数の最上位 16 ビットへアクセスするため使用される。
10	NMSR	デクリメントオンリー(最上位領域以外)	新しい 16 ビット値は、 $M[n+1]$ に書き込むことができる場合に限り $M[n]$ に書き込まれる。NMSR アクセスモードは、常に、32 ビットの倍精度値及びそれ以上(16 ビットの倍数)をデクリメントすることができる。
11	RO	リードオンリー	新しい 16 ビット値は無視される。 $M[n]$ は変えられない。

16 個の $M[n]$ レジスタのための 16 組のアクセスモードビットは、単一のアクセスモード `Access Mode` レジスタにまとめられる。32 ビットの `Access Mode` レジスタは、以下のように、 n 付きの $M[n]$ に対応する。

【1 2 0 5】

【表 1 1 4】

MSB								LSB							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

各 2 ビット値は、上位／下位のフォーマットで記憶される。したがって、 $M[0-5]$ がアクセスモード MSR であり、 $M[6-15]$ がアクセスモード RO であるならば、32 ビット `Access Mode` レジスタは、

1 1 - 1 1 - 1 1 - 1 1 - 1 1 - 1 1 - 1 1 - 1 1 - 1 1 - 1 1 - 0 1 - 0 1 - 0 1 - 0 1 - 0 1 - 0 1

である。

【1 2 0 6】

WR (ライト) コマンドの実行中に、`Access Mode[n]` は各 $M[n]$ について調べられ、新しい $M[n]$ 値が古い方を置き換えるかどうかを判断する。`Access Mode` レジスタは認証チップの SAM (セットアクセスモード) コマンドを使用してセットされる。デクリメントオンリー比較は符号無しであり、負のレンジを要求するデクリメントオンリー値は、正のレンジにシフトさせなければならない。例えば、-50 から 50 のレンジのデクリメントオンリーデータ項目を含む消耗品は、そのレンジが 0 から 100 へシフトされる。システムは、そのレンジ 0 から 100 を -50 から 50 として解釈する。尚、殆どの場合に、デクリメントオンリーレンジは N から 0 であり、レンジシフトは必要ない。デクリメントオンリーデータ項目の場合、データは、最上位 16 ビット量から最

下位16ビット量の順にM[n]から前方へ並べられる。最上位16ビット(M[n]に記憶されている)に対するアクセスモードは、MSRにセットされるべきである。残りのレジスタ(M[n+1]、M[n+2]等)は、それらのアクセスモードがNMSRにセットされるべきである。誤ってNMSRにセットされ、関連したMSR領域が無い場合、各NMSR領域は、倍精度比較ではなく、独立であるとみなされる。

【1207】

K₁

K₁は認証プロトコル中にRを変換するため使用される160ビット秘密鍵である。K₁は、SSI(秘密情報セット)コマンドでK₂及びRと共にプログラミングされる。K₁は秘密にされなければならないので、クライアントはK₁を直接読み出し得ない。K₁を利用するコマンドはRND及びRDである。RNDは、R、F_{K₁}[R]のペアを返し、Rは乱数であり、一方、RDは入力としてX、F_{K₂}[R]のペアを要求する。K₁は鍵付き一方向ハッシュ関数HMAC-SHA1で使用される。したがって、K₁は、物理的にランダムな現象から収集された、物理的に生成された乱数でプログラミングされるべきである。K₁は、決して、コンピュータ上で実行される乱数生成器で生成してはならない。認証チップの安全性は、決定論的では無い方法で生成されたK₁、K₂及びRに依存する。例えば、K₁をセットするため、ある人は公正なコインを160回トスし、表を1として記録し、裏を0として記録する。K₁は、CLRコマンドの実行時に自動的に0にクリアされる。それは、SSIコマンドだけによって非零値にセットすることができる。

【1208】

K₂

K₂は認証プロトコル中にM|Rを変換するため使用される160ビット秘密鍵である。K₂は、SSI(秘密情報セット)コマンドでK₁及びRと共にプログラミングされる。K₂は秘密にされなければならないので、クライアントはK₂を直接読み出し得ない。K₂を利用するコマンドはRD及びTSTである。RDは、M、F_{K₂}[M|X]のペアを返し、Xはパラメータのうちの一つとしてRD関数へ渡される。TSTは入力としてM、F_{K₂}[M|R]のペアを要求し、Rは認証チップのRND関数から獲得された。K₂は鍵付き一方向ハッシュ関数HMAC-SHA1で使用される。したがって、K₂は、物理的にランダムな現象から収集された、物理的に生成された乱数でプログラミングされるべきである。K₂は、決して、コンピュータ上で実行される乱数生成器で生成してはならない。認証チップの安全性は、決定論的では無い方法で生成されたK₁、K₂及びRに依存する。例えば、K₂をセットするため、ある人は公正なコインを160回トスし、表を1として記録し、裏を0として記録する。K₂は、CLRコマンドの実行時に自動的に0にクリアされる。それは、SSIコマンドだけによって非零値にセットすることができる。

【1209】

RとI s T r u s t e d

Rは、SSI(秘密情報セット)コマンドでK₁及びRと共にプログラミングされる160ビットの乱数シードである。Rは秘密にする必要がない。なぜならば、それは、RNDコマンドによって自由に呼び出し元へ与えられるからである。しかし、Rを変更できるのは認証チップだけであり、呼び出し元によって任意の選択した値にセットできない。Rは、TSTコマンド中に使用され、前のRND呼び出しからのRが、信頼できない認証チップ(チップA)においてF_{K₂}[M|R]を生成するために使用されたことを保証する。RND及びTSTの両方は、信頼されている認証チップ(チップT)だけで使用される。I s T r u s t e dは1ビットフラグレジスタであり、認証チップが信頼されているチップ(チップT)であるかどうかを決定する：

I s T r u s t e dビットがセットされているならば、チップは信頼されているチップであるとみなされるので、クライアントはRND関数及びTST関数を呼び出すことができる(RD又はWRではない)；

I s T r u s t e dビットがクリアされているならば、チップは信頼されているチップであるとみなされない。したがって、RND関数及びTST関数を呼び出すことができない

(その代わり、RDとWRを呼び出しても構わない)。システムは、消耗品上のRND又はTSTを決して呼び出す必要がない(なぜならば、複製チップはTSTのような関数に1を返すだけであり、RNDに対して一定値を返すだけである)。

【1210】

IsTrustedビットは、攻撃者によって獲得できる利用可能な R 、 $F_{K_1}[R]$ の個数を減少させる付加的な利点があり、しかも認証プロトコルの完全性を維持する。有効な R 、 $F_{K_1}[R]$ のペアを獲得するため、攻撃者は、システム認証チップが必要であり、これは、消耗品よりも高価であり入手が困難である。 R 及びIsTrustedビットの両方はCLRコマンドによって0にクリアされる。これらは共に、SSIコマンドの発行によって書き込まれる。IsTrustedビットは、SSIコマンドによって、非零シード値を R に格納することによってのみセットすることができる(R は有効なLFSR状態であるために非零であり、これは全く合理的である)。 R は、ビット1、2、4及び159にタップが付いた160ビットの最大限の周期のLFSRによって変更され、TSTへの呼び出しが成功したときだけに(1が返された場合)変更される。

【1211】

システムで信頼されているチップ(チップT)になることが予定されている認証チップは、プログラミング中にそのIsTrustedビットがセットされるべきであり、消耗品で使用される認証チップ(チップA)は(プログラミング中にSSIコマンドで R に0を格納することにより)そのIsTrustedビットがクリアのままにされるべきである。IsTrustedビットを直接読み書きするためのコマンドは存在しない。認証チップの安全性は、 K_1 及び K_2 のランダム性、並びに、HMAC-SHA1アルゴリズムの強度だけに依存するわけではない。攻撃者が疎ルックアップテーブルを構築することを妨げるため、認証チップの安全性は、全てのシステムの耐用期間に亘る R のレンジにも依存する。したがって、 R は、物理的にランダムな現象から収集された、物理的に生成された乱数でプログラミングされるべきである。 R は、決して、コンピュータ上で実行される乱数生成器で生成してはならない。 R の生成は決定論的であってはならない。例えば、信頼できるシステムチップで使用するための R を生成するため、ある人は公正なコインを160回トスし、表を1として記録し、裏を0として記録する。0は信頼できる R に対する唯一の有効でない初期値である(又は、IsTrustedビットはセットされない)。

【1212】

SIWritten

SIWritten(秘密情報書き込み(Secret Information Written))1ビットレジスタは、認証チップ内に記憶された秘密情報の状態を保持する。秘密情報は K_1 、 K_2 及び R である。クライアントはSIWrittenビットに直接アクセスできない。その代わりに、SIWrittenビットはCLRコマンドによってクリアされる(このコマンドは、 K_1 、 K_2 及び R もクリアする)。認証チップが、SSIコマンド(書き込まれる値とは無関係に)を使用して秘密鍵及び乱数シードでプログラミングされたとき、SIWrittenビットは自動的にセットされる。 R は厳密には秘密ではないが、攻撃者が選択された R 、 $F_{K_1}[R]$ のペアを獲得するために自分の固有の乱数シードを生成し得ないことを保証するため、 R は K_1 及び K_2 と一緒に書き込まなければならない。SIWritten状態ビットは、 K_1 、 K_2 又は R にアクセスする全ての関数によって使用される。SIWrittenビットがクリアされている場合、RD、WR、RND及びTSTの呼び出しはCLRの呼び出しとして解釈される。

【1213】

MINTICKS

攻撃者が短時間の間にTST及びRD関数への多数の呼び出しを生成できないようにする二つのメカニズムが存在する。第1のメカニズムは、内部クロックが特定の最大値(例えば、10MHz)を超える速度で動作することを妨げるクロック制限ハードウェアコンポーネントである。第2のメカニズムは32ビットのMinTicks(最小チック)レジスタであり、これは、鍵付き関数への呼び出しの間に経過しなければならない最小のクロ

ックチック回数を指定するため使用される。MinT i c k s 変数はCLRコマンドによって0にクリアされる。ビットは、SMT (MinT i c k s セット) コマンドによってセットされる。SMTへの入力パラメータは、MinT i c k s のどのビットをセットすべきであるかを表現するビットパターンを含む。実際の効果は、攻撃者がMinT i c k s の単位でしか値を増加し得ないことである（なぜならば、SMT関数はビットをセットするだけである）。その上、呼び出し元がこのレジスタの現在の値を読み出すことができる関数は設けられていない。MinT i c k s の値は、動作クロック速度と、鍵付き関数呼び出しの間の合理的な時間の定め方（アプリケーションに固有）と、に依存する。1チックの間隔は動作クロック速度に依存する。これは、入力クロック速度と認証チップのクロック制限ハードウェアの最大値である。例えば、認証チップのクロック制限ハードウェアは10MHzにセットされるが（これは変更できない）、入力クロックは1MHzである。このケースでは、1チックの値は10MHzではなく、1MHzが基準になる。入力クロックが1MHzではなく20MHzであるならば、1チックの値は10MHzが基準になる（なぜならば、クロック速度は10MHzで制限されている）。

10

【1214】

1チックの間隔がわかると、MinT i c k s 値をセットすることができる。MinT i c k s の値は、鍵付きRD及びTST関数への呼び出しの間に経過することが要求されるMinT i c k s 数である。この値は、リアルタイム数であり、動作チックの長さで除算される。入力クロック速度が10MHzの最大クロック速度と一致する場合を想定する。鍵付き関数の呼び出しの間に1秒の最小値を求めるならば、MinT i c k s の値は1000000にセットされる。攻撃者がRND、RD及びTSTを何回も呼び出すことによってX、F_{K1}[X]のペアを収集しようとしている場合を考える。TSTへの呼び出しの間の時間が1秒になるようにMinT i c k s 値がセットされているならば、各ペアを生成するために1秒が必要になる。2²⁵個のペア（1.25Gbの記憶量しか必要にならない）を生成するため、攻撃者は1年以上を必要とする。2⁶⁴個のペアが必要になる攻撃は、1個のチップを使用すると、5.84×10¹¹年が必要になり、10億個のチップが使用されるならば、584年が必要になり、このような攻撃は時間的に全く非現実的になる（記憶必要量については言うまでもない！）。

20

【1215】

K₁に関しては、MinT i c k s 変数は攻撃者を遅くさせ、攻撃のコストを高くするだけである。なぜならば、MinT i c k s は攻撃者が多数のシステムチップを並列に使用することを止めるものではないからである。しかし、MinT i c k s は、本当にK₂への攻撃をより困難にさせる。なぜならば、各消耗品は異なるMを有するからである（Mの一部はランダムなリードオンリーデータである）。差分攻撃に出るためには、最小限度の差の入力が必要であり、これは、単一の消耗品（Mの実効的に定数部を含む）によって実現できる。最小限度の差の入力は、攻撃者が単一チップを使用することを要求し、MinT i c k s は、単一チップの使用をスローダウンさせる。差分攻撃を始めるために値の検索を開始するためのデータを取得するだけに1年を要するのであれば、これは攻撃のコストを上昇させ、複製消耗品の実効的な販売期間を短縮する。

30

【1216】

認証チップコマンド

システムは簡単なオペレーションコマンドセットによって認証チップと通信する。このセクションは、プロトコル3の実装のために必要な実際のコマンド及びパラメータを詳述する。認証チップは、最低限の実装としてシリアルインタフェースによってシステムと通信するものとして定義される。より幅広いインタフェース（例えば、8、16又は32ビット）によって動作する等価的なチップを定義することは慣用的なことである。オペコードの解釈は、IsTrustedビットの現在の値と、IsWrittenビットの現在の値と、に依存し得る。以下のオペレーションが定義される。

40

【1217】

【表 1 1 5】

Op	T	W	Mn	入 力	出 力	説 明
000	—	—	CLR	—	—	クリア
001	0	0	SSI	[160, 160, 160]	—	秘密情報セット
010	0	1	RD	[160, 160,]	[256, 160,]	Mを安全に読み出し
010	1	1	RND	—	[160, 160,]	ランダム
011	0	1	WR	[256]	—	Mを書き込み
011	1	1	TST	[256, 160,]	[1]	テスト
100	0	1	SAM	[32]	[32]	AccessMode セット
101	—	1	GIT	—	[1]	IsTrausted 獲得
110	—	1	SMT	[32]	—	MinTicks セット

10

Op = オペコード、T = Is Trusted 値、W = Is Written 値、
Mn = ニューモニック、[n] = パラメータに必要なビット数。

【1 2 1 8】

この表に定義されていないコマンドは、NOP（ノーオペレーション）として解釈される。その例には、（Is Trusted 値又は Is Written 値とは無関係である）オペコード 110 及び 111 と、Is Written = 0 のときの SSI 以外のオペコードと、が含まれる。RD 及び RND のオペコードは同じであり、WR 及び TST のオペコードも同じであることに注意する必要がある。オペコードの受け取り後に動かされる実際のコマンドは、（Is Written が 1 である限り）Is Trusted ビットの現在値に依存する。Is Trusted ビットがクリアされているところでは、RD 及び WR 関数が呼び出される。Is Trusted ビットがセットされると、RND 及び TST 関数が呼び出される。2 組のコマンドは、信頼されている認証チップと信頼されていない認証チップとの間で相互に排他的であり、同じオペコードはこの関係を強いる。各コマンドは引き続きセクションで詳細に検討される。一部のアルゴリズムは具体的に設計される。なぜならば、フラッシュメモリが不揮発変数の実現形態として想定されるからである。

20

【1 2 1 9】

30

【表 1 1 6】

CLR	クリア
入力	無し
出力	無し
変化	全部

CLR（クリア）コマンドは、全ての認証チップメモリの内容を完全に消去するため設計される。これは、全ての鍵及び秘密情報と、アクセスモードビットと、状態データと、を含む。CLR コマンドの実行後、認証チップは、新しく製造されたときと丁度同じように、プログラマブル状態になる。それは、新しい鍵として再プログラムして、再使用することができる。CLR コマンドは、CLR コマンドオペコードだけにより構成される。認証チップはシリアルであるため、これは同時に 1 ビットずつ転送される。ビット順序は、各コマンドコンポーネントの LSB から MSB である。したがって、CLR コマンドは、CLR オペコードのビット 0 から 2 として送られる。全部で 3 ビットが転送される。CLR コマンドはいつでも直ちに呼び出すことができる。消去の順序は重要である。SIWritten bit 値は、（RND、TST、RD 及び WR のような）鍵アクセス関数への更なる呼び出しを禁止するため、最初に消去されなければならない。AccessMode

40

50

ビットが `SIWritten` の前にクリアされると、攻撃者は、それらがクリアされた後のある時点で電源を取り外し、`M` を操作し、これにより、部分選択テキスト攻撃で秘密情報を取り出すよい機会を得る。`CLR` コマンドは以下のステップで実現される。

【1220】

【表117】

ステップ	動作
1	<code>SIWritten</code> 消去 <code>IsTrusted</code> 消去 K_1 消去 K_2 消去 <code>R</code> 消去 <code>M</code> 消去
2	<code>AccessMode</code> 消去 <code>MinTicks</code> 消去

チップがクリアされると、チップは再プログラミングと再使用の準備が完了する。ブランクチップは攻撃者にとって無用である。なぜならば、`M` に対して任意の値を作成することが可能であるとしても（`M` は読み出し及び書き込みが可能である）、 K_1 及び K_2 が間違っているので鍵付き関数は全く情報を提供しないからである。`CLR` が `CLR` 以外の任意のオペコードのために要求されるならば、入力パラメータビットを費やす必要がない。攻撃者はチップをリセットするだけでよい。`CLR` を呼び出す理由は、全ての秘密情報が破壊され、チップが攻撃者にとって無用になることを保証するためである。

【1221】

`SSI`—秘密情報セット (`SET SECRET INFORMATION`)

入力： K_1 、 K_2 、`R` = [160ビット, 160ビット, 160ビット]

出力：無し

変更内容： K_1 、 K_2 、`R`、`SIWritten`、`IsTrusted`。

【1222】

`SSI` (秘密情報セット (`Set Secret`

`Information`)) コマンドは、 K_1 、 K_2 及び `R` 変数にロードし、`RND`、`ST`、`RD` 及び `WR` コマンドを後で呼び出すため `SIWritten` 及び `IsTrusted` フラグをセットするため使用される。`SSI` コマンドは、`SSI` コマンドオペコードと、その後続く、 K_1 、 K_2 及び `R` レジスタに格納されるべき秘密情報と、を含む。認証チップはシリアルであるため、これは同時に1ビットずつ転送されるべきである。ビット順序は、各コマンドコンポーネントのLSBからMSBである。したがって、`SSI` コマンドは、`SSI` オペコードのビット0-2と、その後続く、 K_1 に対する新しい値のビット0-159と、 K_2 に対する新しい値のビット0-159と、最後に、`R` に対するシード値のビット0-159として送られる。全部で483ビットが転送される。 K_1 、 K_2 、`R`、`SIWritten`、及び `IsTrusted` レジスタは `CLR` コマンドによって全て0にクリアされる。それらは、`SSI` コマンドを使用することによってのみセットすることができる。

【1223】

`SSI` コマンドは、データが K_1 、 K_2 及び `R` にロードされた事実を格納するためフラグ `SIWritten` を使用する。`SIWritten` フラグ及び `IsTrusted` フラグをクリアされているならば（これは、`CLR` 命令後のケースである）、 K_1 、 K_2 及び `R` に新しい値がロードされる。何れかのフラグがセットされているならば、`SSI` の呼び出しの試みによって、`CLR` コマンドが実行される。なぜならば、攻撃者又は間違ったクライアントだけが、`CLR` を先に呼び出すことなく、鍵又はランダムシードを変更しようとするからである。また、`SSI` コマンドは、`R` に対する値に依存して `IsTruste`

dフラグをセットする。R = 0であるならば、チップは信用できないとみなされ、したがって、IsTrustedは0のまま維持される。R ≠ 0であるならば、チップは信用できるとみなされ、IsTrustedは1にセットされる。IsTrustedビットの設定は、SSIコマンドの間だけで行われることに注意する必要がある。認証チップが再使用されるならば、CLRコマンドは最初に呼び出されなければならない。その後、鍵は、SSIコマンドで安全にプログラムされ、新たな状態情報がSAM及びWRコマンドを用いてMへロードされる。SSIコマンドは以下のステップで実行される。

【1224】

【表118】

ステップ	動作
1	CLR
2	$K_1 \leftarrow$ クライアントから読み出した 160 ビット
3	$K_2 \leftarrow$ クライアントから読み出した 160 ビット
4	$R \leftarrow$ クライアントから読み出した 160 ビット
5	IF($R \neq 0$) IsTrusted ← 1
6	SIWritten ← 1

RD－読み出し

入力：X、 F_{K1} [X] = [160ビット、160ビット、160ビット]

出力：M、 F_{K2} [X | M] = [256ビット、160ビット]

変更内容：R。

【1225】

RD（リード）コマンドは、信頼されない認証チップから状態データ（M）の256ビット全体を安全に読み出すために使用される。有効認証チップだけがRD要求に正しく応答する。RDコマンドからの出力ビットは、入力ビットとして、検証用の信頼された認証チップ上のTSTコマンドへ供給でき、最初の256ビット（M）は、TSTが（期待通りに）1を戻す場合に後で使用するため保存される。認証チップはシリアルであるため、コマンド及び入力パラメータは同時に1ビットずつ転送されるはずである。ビット順序は、各コマンドコンポーネントのLSBからMSBである。したがって、RDコマンドは、RDオペコードのビット0－2と、その後続く、Xのビット0－159と、 F_{K1} [X]のビット0－159である。全部で320ビットが転送される。X及び F_{K1} [X]は信頼された認証チップのRNDコマンドを呼び出すことによって獲得される。信頼されたチップのRNDコマンドから出力された320ビットは、したがって、信頼されないチップのRDコマンドへそのまま供給することが可能であり、それらのビットをシステムによって保存しなくてもよい。RDコマンドは、以下の条件が充たされたときに限り使用できる：

SIWritten = 1 K_1 、 K_2 、RがSSIコマンドによってセットアップされたことを示す；

IsTrusted = 0 乱数シーケンスを生成することが許可されないのでチップは信頼されないことを示す。

【1226】

加えて、RDの呼び出しは、MinTicksRemainingレジスタが0に達するまで待たなければならない。そうなった後、レジスタには、MinTicksが再ロードされ、RDへの呼び出しの間に最小限度の時間が確実に経過する。MinTicksRemainingにMinTicksが再ロードされると、RDコマンドは入力パラメータが有効であることを検証する。これは、入力Xに対する F_{K1} [X]を内部的に生成し、次に、得られた結果を入力 F_{K1} [X]と比較することによって行われる。この生成及び比較は、入力パラメータが正しいかどうかとは無関係に同じ時間を使用しなければならない

い。時間が同一ではない場合、攻撃者は、 $F_{K1}[X]$ のどのビットが間違っているかに関する情報を得ることができる。入力パラメータが無効になるのは、間違っただシステム（誤ったビットを渡す）、間違っただシステムに間違っただ消耗品がある場合、不良の信頼されるチップ（不良なペアを生成する）、又は認証チップへの攻撃に限られる。一定値 0 は入力パラメータが間違っている場合に返される。0 が返されるまでに要する時間は、あらゆる不良な入力に対して同じでなければならないので、攻撃者は何が無効であるかに関する情報を全く学習できない。入力パラメータが検証されると、出力値が計算される。M の 256 ビットの内容は、 $M[0]$ のビット 0-15、 $M[1]$ のビット 0-15 から、 $M[15]$ のビット 0-15 の順番に転送される。 $F_{K2}[X|M]$ は計算され、ビット 0-159 として出力される。レジスタは、X、 $F_{K1}[X]$ ペアの検証中に、値を格納するため使用される。その理由は、RND と RD が相互に排他的であるためである。RD コマンドは以下のステップで実施される。

【1227】

【表 119】

ステップ	動作
1	IF (MinTicksRemaining $\neq$ 0) GOTO 1
2	MinTicksRemaining $\leftarrow$ MinTicks
3	R $\leftarrow$ クライアントから読み出した 160 ビット
4	Hash $\leftarrow F_{K1}[R]$
5	OK $\leftarrow$ (Hash = クライアントからの次に 160 ビット) この演算は、アタッカーが自分の推測がどの程度正確であるかを判定できないように一定時間を要することに注意する必要がある。
6	IF (OK) 256 ビットの M をクライアントへ出力 ELSE 256 ビットの 0 をクライアントへ出力
7	Hash $\leftarrow F_{K2}[R M]$
8	IF (OK) 160 ビットの Hash をクライアントへ出力 ELSE 160 ビットの 0 をクライアントへ出力

RND-ランダム

入力：無し

出力：R、 $F_{K1}[R] = [160 \text{ ビット}, 160 \text{ ビット}]$

変更内容：無し。

【1228】

RND（ランダム）コマンドは、RD 及び TST コマンドによって、後続の認証で使用するための有効 R、 $F_{K1}[R]$ ペアを獲得するためクライアントによって使用される。入力パラメータが無いので、RND コマンドは単に RND オペコードのビット 0-2 である。RND コマンドは以下の条件が満たされた場合に限り使用できる：

SIWritten = 1 K_1 および R が SSI コマンドによってセットアップされたことを示す；

Isttrusted = 1 チップは乱数シーケンスを生成することが許可されていることを示す。

【1229】

RND は、R と $F_{K1}[R]$ の両方を呼び出し元へ返す。RND コマンドの 288 ビット出力は、信頼されないチップの RD コマンドに入力パラメータとして供給することが可能

である。クライアントはそれらを全く格納する必要が無い。なぜならば、それらは再度要求されないからである。しかし、T S T コマンドは、R D コマンドに渡された乱数が最初に R N D コマンドから取得されたものである場合に限り成功する。呼び出し元が R N D だけを複数回呼び出すならば、同じ R、 $F_{K-1}[R]$ のペアが毎回返されるであろう。R は、T S T への呼び出しが成功であった後、シーケンス内の次の乱数へ進む。より詳しくは T S T の説明を参照せよ。R N D コマンドは以下のステップで実施される。

【1 2 3 0】

【表 1 2 0】

ステップ	動 作
1	160 ビットの R をクライアントへ出力
2	Hash ← Calculate $F_{K1}[R]$
3	160 ビットの Hash をクライアントへ出力

10

T S T - テスト (T E S T)

入力：X、 $F_{K-2}[R | X] = [256 \text{ ビット}, 160 \text{ ビット}]$

出力：1 又は 0 = [1 ビット]

変更内容：M、R 及び M i n T i c k s R e m a i n i n g (又は攻撃が検出されたならば全てのレジスタ)。

20

【1 2 3 1】

T S T (テスト (T e s t)) コマンドは、信頼されない認証チップからの M の読み出しを認証するため使用される。T S T (テスト) コマンドは、T S T コマンドオペコードと、その後に続く入力パラメータ：X 及び $F_{K-2}[R | X]$ と、により構成される。認証チップはシリアルであるため、これは同時に 1 ビットずつ転送されなければならない。ビット順序は、各コマンドコンポーネントの L S B から M S B である。したがって、T S T コマンドは、T S T オペコードのビット 0-2 と、その後に続く、M のビット 0-255 と、 $F_{K-2}[R | M]$ のビット 0-159 である。全部で 419 ビットが転送される。最後の 416 ビットは、R D コマンドから信頼されない認証チップへの出力ビットとして獲得されるので、データ全体をクライアントが保存する必要はない。その代わりに、ビットは、信頼された認証チップの T S T コマンドへそのまま渡すことができる。M のうちの 256 ビットだけが R D コマンドに秘密にされるべきである。T S T コマンドは、以下の条件が満たされたときに限り使用できる：

30

S I W r i t t e n = 1 K_2 および R が S S I コマンドによってセットアップされたことを示す；

I s T r u s t e d = 1 チップは乱数シーケンスを生成することが許可されていることを示す。

【1 2 3 2】

加えて、T S T の呼び出しは、M i n T i c k s R e m a i n i n g レジスタが 0 に達するまで待たなければならない。そうなった後、レジスタには、M i n T i c k s が再ロードされ、T S T への呼び出しの間に最小限度の時間が確実に経過する。T S T は、内部 M 値を入力 M 値によって置き換えさせる。次に、 $F_{K-2}[M | R]$ が計算され、160 ビット入力ハッシュ値と比較される。生成される単一出力ビットは、両者が同じであるならば 1 であり、両者が異なるならば 0 である。内部 M 値を使用することは、チップ上のスペースを節約し、R D 及び T S T が相互に排他的なコマンドであること理由である。出力ビットが 1 であるならば、R はシーケンス内の次の乱数になるように更新される。これは、R D 及び T S T テストが呼び出される毎に、呼び出し元に強制的に新しい乱数を使用させる。結果の出力ビットは、入力文字列全体が比較されるまでは出力されないため、T S T 関数内の比較を評価するために要する時間は常に同じである。このようにして、攻撃者は、実行時間、又は出力が与えられる前に処理されたビット数を比較できない。

50

【1233】

次の乱数は、160ビットの最大周期のLFSR（タップはビット159、4、2及び1で選択）を使用してRから生成される。Rの初期160ビット値はSSIコマンドによってセットアップされ、0以外のいかなる乱数でもよい（0で満たされたLFSRは際限のない0のストリームを生成する）。Rはビット1、2、4及び159をまとめてXOR論理演算し、XOR論理演算の結果を b_{159} への入力として用いて160ビット全部を右へ1ビットシフトすることにより変換される。新しいRは次のRND呼び出しの際に返される。TSTから0を返すために要する時間は、全ての不良な入力に対して同じであり、攻撃者は無効な入力に関して何も学習し得ないことに注意する必要がある。

【1234】

TSTコマンドは以下のステップで実施される。

【1235】

【表121】

ステップ	動作
1	IF (MinTicksRemaining $\neq$ 0) GOTO 1
2	MinTicksRemaining $\leftarrow$ MinTicks
3	M $\leftarrow$ クライアントから読み出した 256 ビット
4	IF (R = 0) GOTO CLR
5	Hash $\leftarrow$ Calculate $F_{K2}[R \parallel M]$
6	OK $\leftarrow$ (Hash = クライアントからの次の 160 ビット) この演算は一定時間を要するので、アタッカーは自分の推測がどの程度正しいかを判定できない。
7	IF (OK) Temp $\leftarrow$ R 消去 R LFSR によって TEMP を前進 R $\leftarrow$ TEMP
8	OK の 1 ビットをクライアントへ出力

尚、ステップ7では、単純にそのままRを進めることができない。なぜならば、Rはフラッシュメモリであり、セットされたビットが0になるためには消去しなければならないからである。電源が、古いRの値を消去した後で新しいRの値が書き込まれる前に、ステップ7の間に認証チップから取り外された場合、Rは消去されるが再プログラミングされない。したがって、IsTrusted=1でありながら、R=0であるという状況が生まれ、この状況が起こり得るのは攻撃者が原因となる場合だけである。ステップ4はこの事象を検出し、攻撃が検出された場合に動作する。この問題は、Rのための第2の160ビットフラッシュレジスタと有効性ビットとを設け、新しい値がロードされた後にトグル切替することで防ぐことができる。これは、スペースの理由で本実施形態には組み込まれていないが、チップ空間がそれを許容するならば、補助的な160ビットフラッシュレジスタがこの目的のために有用であろう。

【1236】

WR－書き込み (WRITE)

入力: $M_{new} = [256 \text{ ビット}]$

出力: 無し

変更内容: M。

【1237】

WR (書き込み (Write)) コマンドは、認証チップ状態データを格納するMの書き

10

20

30

40

50

込み可能な部分を更新するため使用される。WR コマンド自体は安全ではない。WR コマンドの後には、(RD コマンドによる) 認証されたMの読み出しが続き、指定されたとおりに変更がなされたことを保証する。WR コマンドは、WR コマンドオペコードと、その後続くMへ書き込まれるべき新しい256ビットのデータと、を渡すことによって呼び出される。認証チップはシリアルであるため、Mの新しい値は同時に1ビットずつ転送されなければならない。ビット順序は、各コマンドコンポーネントのLSBからMSBである。したがって、WR コマンドは、WR オペコードのビット0-2と、その後続く、M[0]のビット0-15と、M[1]のビット0-15からM[15]のビット0-15までである。全部で259ビットが転送される。WR コマンドは、SIWritten=1のとき、即ち、 K_1 、 K_2 、RがSSI コマンドによってセットアップされたことを示すときに限り使用できる(SIWrittenが0であるならば、 K_1 、 K_2 、Rは未だセットアップされていないので、代わりにCLR コマンドが呼び出される)。特定のM[n]への書き込みの可能性は、AccesssModeレジスタに格納されているような対応したアクセスモードビットにより支配される。AccesssModeビットはSAM コマンドを用いてセットできる。新しい値をM[n]へ書き込みとき、M[n]はフラッシュメモリであるということを考慮しなければならない。M[n]の全てのビットを消去した後、適切なビットをセットしなければならない。二つのステップが別々のサイクルで行われるので、攻撃の可能性は無防備のままである。攻撃者は、消去後、しかし、新しい値をプログラミングする前に、電源を取り外すことが可能である。しかしながら、攻撃者はこれを行っても利益がない：

この手段によってリード/ライトM[n]が0に変更されることは全く利益がない。なぜならば、攻撃者はWR コマンドを使用して任意の値を書き込めるからである；

この手段によってリードオンリーM[n]が0に変更されることによって、付加的なテキストペアを知ることができる(ここで、M[n]は元の値ではなく0である)。将来M[n]値を使用する場合、それらは既に0であるため、全く情報が与えられない；

デクリメントオンリーM[n]が0に変更されると、消耗品が使い切られる時間が早まるだけである。これは、消耗品を使用することによって得られるような何ら新しい情報を攻撃者に与えることはない。

【1238】

WR コマンドは以下のステップで実施される。

【1239】

10

20

30

【表 1 2 2】

ステップ	動 作
1	DecEncountered←0 EqEncountered←0 n←15
2	Temp←クライアントから読み出した 16 ビット
3	AM = AccessMode [-n]
前の値と比較	
5	LT← (Temp < M [-n]) [比較は符号無し] EQ← (Temp = M [-n])
6	WE← (AM = RW) ∨ ((AM=MSR) ∧ LT) ∨ ((AM=NMSR) ∧ DecEncountered ∨ LT))
7	DecEncountered← ((AM=MSR) ∧ LT) ∨ ((AM=NMSR) ∧ DecEncountered ∨ ((AM=NMSR) ∧ EqEncountered ∧ LT)) EqEncountered← ((AM=MSR) ∧ EQ) ∨ ((AM=NMSR) ∧ EqEncountered ∧ EQ)
次のアクセスモードセットへ進 み、適用できるならば新しい値 M [-n]を書き込む	
8	IF (WE) Erase M [-n] M[-n]←Temp
10	↓ _n
11	IF(n ≠ 0) GOTO 2

10

20

30

40

50

S A Mーアクセスモードセット (S E T A C C E S S M O D E)

入力: A c c e s s M o d e _ n _ e w = [3 2 ビット]

出力: A c c e s s M o d e = [3 2 ビット]

変更内容: A c c e s s M o d e。

【 1 2 4 0】

S A M (アクセスモードセット (S e t A c c e s s M o d e)) コマンドは、A c c e s s M o d e レジスタの 3 2 ビットをセットするため使用され、消耗品認証チップでの使用だけが有効である (I s T r u s t e d フラグ = 0 の場合)。S A M コマンドは、S A M コマンドオペコードと、その後続く、A c c e s s M o d e レジスタにビットをセットするために使用される 3 2 ビット値と、を渡すことによって呼び出される。認証チップはシリアルであるため、データは同時に 1 ビットずつ転送されなければならない。ビット順序は、各コマンドコンポーネントの L S B から M S B である。したがって、S A M コマンドは、S A M オペコードのビット 0 - 2 と、その後続く、A c c e s s M o d e にセットされるビットのビット 0 - 3 1 と、である。全部で 3 5 ビットが転送される。A c c e s s M o d e レジスタは C L R コマンドの実行時に限り 0 にクリアされる。0 0 というアクセスモードは R W (リード/ライト) のアクセスモードを示すので、C L R 後に A c c e s s M o d e ビットをセットしないことは、全ての M が読み書きできることを意味する。S A M コマンドだけが A c c e s s M o d e レジスタのビットをセットする。したがって、クライアントは、3 2 ビット語に適当なビットをセットし、入力パラメータとしての 3 2 ビット値を用いて S A M を呼び出すことにより、M [n] に対するアクセスモードビットを R W から R O (リードオンリー) へ変更可能である。これは、様々な時点での

、おそらく製造プロセスの種々のステージにおけるアクセスモードビットのプログラミングを可能にさせる。例えば、リードオンリーランダムデータは、初期鍵プログラミングステージ中に書き込むことが可能であり、一方、消耗品シリアル番号のような項目のための第2のプログラミングステージが許可される。

【1241】

SAMコマンドはビットをセットするだけであるので、その効果は、M[n]に対応するアクセスモードビットをRWからMSR、NMSR又はROの何れかへ進行させ得ることである。尚、MSRのアクセスモードはROへ変更可能であるが、これは、攻撃者には役立たないことに注意する必要がある。なぜならば、不正に加工された認証チップへの書き込み後のMの認証は、書き込みが不成功であったことを検出し、オペレーションを中止するからである。ビットの設定は、フラッシュメモリが最も巧く機能する方法に対応する。AccessModeレジスタ内のビットをクリアする唯一の方法は、例えば、デクリメントオンリーM[n]をリード／ライトに変更するための唯一の方法は、CLRコマンドを使用することである。CLRコマンドは、AccessModeレジスタを消去（クリア）するだけでなく、鍵及び全てのMをクリアする。このようにして、M[n]に対応するAccessMode[n]ビットは、CLRコマンドの間に1回だけ有効に変更することができる。SAMコマンドは（適切なビットが入力パラメータによってセットされた後）AccessModeレジスタの新しい値を返す。入力パラメータを0としてSAMを呼び出すことにより、AccessModeは変更されず、したがって、AccessModeの現在値が呼び出し元へ返される。

【1242】

SAMコマンドは以下のステップで実施される。

【1243】

【表123】

ステップ	動 作
1	Temp←クライアントから読み出した32ビット
2	SetBits(AccessMode, Temp)
3	AccessModeの32ビットをクライアントへ出力

GIT-IsTrustedを取得 (GET ISTRUSTED)

入力：無し

出力：IsTrusted = [1ビット]

変更内容：無し。

【1244】

GIT(IsTrusted取得 (Get IsTrusted) コマンドは、認証チップ上のIsTrustedビットの現在値を読むため使用される。戻されるビットが1であるならば、認証チップは信頼できるシステム認証チップである。戻されるビットが0であるならば、認証チップは消耗品認証チップである。GITコマンドは、GITコマンドオペコードだけを含む。認証チップはシリアルであるため、これは同時に1ビットずつ転送されなければならない。ビット順序は、各コマンドコンポーネントのLSBからMSBである。したがって、SAMコマンドは、SAMオペコードのビット0-2として送信される。全部で3ビットが転送される。GITコマンドは以下のステップで実施される。

【1245】

【表124】

ステップ	動 作
1	IsTrustedビットをクライアントへ出力

SMT-MinTicksセット (SET MINTICKS)

入力：MinTicks_{new} = [32ビット]

出力：無し

変更内容：MinTicks。

【1246】

SMT (MinTicksセット (Set MinTicks)) コマンドは、MinTicksレジスタのビットをセットし、TST及びRDへの呼び出しの間に経過すべきMinTicks数を定めるため使用される。SMTコマンドは、SMTコマンドオペコードと、その後に続く、MinTicksレジスタにビットをセットするために使用される32ビット値と、を渡すことによって呼び出される。認証チップはシリアルであるため、データは同時に1ビットずつ転送されなければならない。ビット順序は、各コマンドコンポーネントのLSBからMSBである。したがって、SMTコマンドは、SMTオペコードのビット0-2と、その後に続く、MinTicksにセットされるビットのビット0-31と、である。全部で35ビットが転送される。MinTicksレジスタはCLRコマンドの実行時に限り0にクリアされる。0という値は、鍵付き関数の呼び出しの間にチックを経過させなくてもよいことを示す。したがって、この関数は、クロック速度制限ハードウェアがチップに動作を許可する頻度で呼び出される。

10

【1247】

SMTコマンドはビットをセットするだけなので、その効果は、クライアントが値をセットできるようにさせ、更なる呼び出しが行われた場合に時間遅延を増大させるだけである。既にセットされたビットをセットすることは効果がなく、クリアされているビットをセットすることだけが、チップを更に遅くさせるために作用する。ビットの設定はフラッシュメモリが最も巧く機能する方法に対応する。MinTicksレジスタ内のビットをクリアする唯一の方法は、例えば、10チックの値を4チックの値に変更するための唯一の方法は、CLRコマンドを使用することである。しかし、CLRコマンドは、MinTicksレジスタを0にクリアするだけでなく、全ての鍵及びMをクリアする。したがって、それは攻撃者にとっては役に立たない。このようにして、MinTicksレジスタは、CLRコマンドの間に1回だけ有効に変更することができる。

20

【1248】

SMTコマンドは以下のステップで実施される。

【1249】

30

【表125】

ステップ	動作
1	Temp←クライアントから読み出した32ビット
2	SetBits (MinTicks, Temp)

認証チップのプログラミング

認証チップは物理的に安全な環境内で論理的に安全な情報でプログラミングしなければならない。その結果として、プログラミング手続は論理的セキュリティと物理的セキュリティの両方を取り扱う。論理的セキュリティは、 K_1 、 K_2 、R及び乱数 $M[n]$ の値が、コンピュータではなく、物理的にランダムなプロセスによって生成されることを保証するプロセスである。また、それは、チップの部品のプログラミングされる順序が最も論理的に安全であることを保証するプロセスでもある。物理的セキュリティは、プログラミングステーションが物理的に安全であり、したがって、鍵生成ステージ期間と鍵の記憶の耐用期間の両方で、 K_1 及び K_2 が秘匿されることを保証するプロセスである。その上、プログラミングステーションは、鍵を獲得又は破壊する物理的攻撃に対抗しなければならない。認証チップは、 K_1 及び K_2 が秘匿されることを保証するための固有のセキュリティメカニズムを備えているが、プログラミングステーションは K_1 及び K_2 を安全に保持しなければならない。

40

【1250】

50

概要

製造後、認証チップは使用可能にされる前にプログラミングする必要がある。全てのチップにおいて、 K_1 及び K_2 の値を確定しなければならない。チップがシステム認証チップを目的としているならば、 R の初期値を決めなければならない。チップが消耗品認証チップを目的としているならば、 R は 0 にセットし、 M 及び $Access Mode$ の初期値も設定しなければならない。したがって、以下のステージが確認される：

システムと消耗品の間の繰り返しを決定する；

システム及び消耗品用の鍵を決定する；

システム及び消耗品用の $MinTicks$ を決定する；

鍵、ランダムシート、 $MinTicks$ 及び未使用 M をプログラムする；

状態データ及びアクセスモードをプログラムする。

10

【1251】

消耗品又はシステムが必要でなくなると、取り付けられた認証チップは再使用可能である。これは、もう一度ステージ 4 から始めて、チップを再プログラミングすることによって簡単に実現できる。各ステージは以下のセクションで説明する。

【1252】

ステージ 0：製造

認証チップの製造は特別なセキュリティを必要としない。製造ステージでは、チップにプログラミングされている秘密情報はない。アルゴリズム及びチッププロセスは特別ではない。標準的なフラッシュプロセスが使用される。チップ製造者とプログラミングステーションの間での認証チップの盗難は、複製品製造者にブランクチップを与えることになるだけである。これは、認証チップの販売を危うくすることは殆どなく、認証チップによって何も認証されない。プログラミングステーションは、消耗品及びシステムのプロダクト鍵を含む唯一のメカニズムであるため、複製品製造者は、正しい鍵を用いてチップをプログラムできない。複製品製造者は、自分専用のシステム及び消耗品用にブランクチップをプログラミングすることは可能であるが、それらの品を見破られずに販売することは困難であろう。その上、1 回の盗みではビジネスの基礎を置くことは難しいであろう。

20

【1253】

ステージ 1：システムと消耗品の間の相互作用の決定

システムとは何か、及び消耗品とは何か、の判断は、認証チップのプログラミングが可能になる前に決定されるべきである。どの消耗品がどのシステムで使用可能であるかに関する判断をする必要がある。なぜならば、接続されるシステムと消耗品だけが同じ鍵情報を共有しなければならないからである。また、それらは、状態データの一部の解釈が未決定であっても、状態データ使用メカニズムを共有すべきである。簡単な例は、車と、車キーの場合である。車自体はシステムであり、車キーは消耗品である。各車には数個の車キーがあり、各キーは特定の車と同じ鍵情報を格納する。しかし、各車（システム）は（その車キーと共有される）異なる鍵をもつであろう。なぜならば、ある車の車キーが別の車で機能することが求められないからである。別の例は、特定のトナーカートリッジを必要とするコピー機の場合である。簡単に言えば、コピー機はシステムであり、トナーカートリッジは消耗品である。しかし、カートリッジとコピー機の間にはどのような互換性があるかを判断する必要がある。この判断は、従来、トナーカートリッジの物理的パッケージングによって行われ、ある種のカートリッジは、新しいモデルのコピー機的设计判断に基づいて、その新しいモデルのコピー機に適合し、或いは、適合しない。認証チップが使用されるとき、連動すべきコンポーネントは同じ鍵情報を共有しなければならない。

30

40

【1254】

更に、消耗品はタイプ毎に、 M （状態データ）の異なる分割方法が必要になる。 M の使用法は、アプリケーション毎に変化するものであり、 $M[n]$ 及び $Access Mode[n]$ の割付方法も同様である：

特定用途の消耗品状態データを定義する；

将来の使用のため（必要に応じて）一部の $M[n]$ レジスタを確保する。それらを 0 にセ

50

ットし、リードオンリーにセットする。値は互換性を保つためにシステムでテストすることができる；

残りのM[n]レジスタ（少なくとも1個、しかし、M[15]であってはならない）をリードオンリーにセットし、各M[n]の内容を完全にランダムにする。これによって、複製品製造者は認証鍵への攻撃がより困難になる；

以下の例は、状態データが編成される方法を示す。

【1255】

例1

車キーが付随した車を想定する。16ビット鍵番号は、所定の車の車キーを一意に識別するために十分足りる。Mの256ビットは以下の通り分割される。

【1256】

【表126】

M[n]	アクセス	説明
0	RO	鍵番号 (16ビット)
1 から 4	RO	カーエンジン番号 (64ビット)
5 から 8	RO	将来の拡張用 = 0 (64ビット)
8 から 15	RO	ランダムビットデータ (128ビット)

車製造者が全ての車の全ての論理的鍵を保持するならば、車キーを紛失したときに、新しい物理的な車キーを製造することは些細な問題である。新しい車キーは、M[0]に新しい鍵番号を格納するが、その車の認証チップと同じ K_1 及び K_2 を含む。車システムは、特定の鍵番号を無効にすることができる（例えば、鍵を紛失したとき）。このようなシステムは、鍵0（マスター鍵）を最初に挿入し、次に全ての有効な鍵を挿入し、次に、鍵0をもう一度挿入することを要求するかもしれない。これらの有効な鍵だけが車と連動するようになる。最悪のケースでは、例えば、全ての車キーを紛失すると、その車に対する新しい論理的な鍵の組を生成し、必要に応じて、関連した物理的な車キーを生成することができる。カーエンジン番号は、その鍵を特定の車に結びつけるために使用される。将来用データは、レンタル情報、例えば、ドライバ／貸出者の詳細情報のような事項を格納する。

【1257】

例2

10000回のコピー毎に交換しなければならないコピー機画像ユニットの例を想定する。残りページ数を記憶するため32ビットが必要になる。Mの256ビットは次の通り分割される。

【1258】

【表127】

M[n]	アクセス	説明
0	RO	シリアル番号 (16ビット)
1	RO	バッチ番号 (16ビット)
2	MSR	残りページカウンタ (32ビット、ハイ／ロー)
3	NMSR	
4 から 7	RO	将来の拡張用 = 0 (64ビット)
8 から 15	RO	ランダムビットデータ (128ビット)

10000回のコピー毎に交換しなければならない低品質画像ユニットが製作されたとき、32ビットのページカウンタは、既存コピー機との互換性のためにそのまま使用される。これにより、数タイプの消耗品を同じシステムで使用できるようになる。

【 1 2 5 9 】

例 3

2 5 枚の写真を格納するポラロイド型カメラの例を考える。残り写真数を記憶するためには 1 6 ビットのカウンタダウンだけあればよい。M の 2 5 6 ビットは次の通り分割される。

【 1 2 6 0 】

【表 1 2 8 】

M[n]	アクセス	説 明
0	RO	シリアル番号 (16ビット)
1	RO	バッチ番号 (16ビット)
2	MSR	残り写真数 (16ビット)
3 から 6	RO	将来の拡張用 = 0 (64ビット)
7 から 15	RO	ランダムビットデータ (144ビット)

10

M [2] の残り写真数は、同じカメラシステムと使用するための多数の消耗品タイプの構築を可能にさせる。例えば、写真数 3 6 の新しい消耗品は簡単にプログラムできる。カメラの発表後、2 年が経過した場合を考えると、新しいタイプのカメラが発表されている。それは、古い消耗品を使用することが可能であるが、新しいフィルムタイプも処理できる。M [3] はフィルムタイプを定義するため使用できる。古いフィルムタイプは 0 であり、新しいフィルムタイプは他の新しい値である。新しいシステムはこれを利用することができる。元のシステムは M [3] に非零値を検出し、新しいフィルムタイプとの非互換性を認識する。新しいシステムは、M [3] の値を理解し、適切に応じる。古い消耗品との互換性を保つため、新しい消耗品及びシステムは、古いものと同じ鍵情報をもつことが必要である。新しいシステム及びその専用消耗品と完全に断ち切るためには、新しい鍵セットが必要になるであろう。

20

【 1 2 6 1 】

例 4

シアン、マゼンタ、及びイエローの三つのインクを格納するプリンタ消耗品の例を考える。各インク量は別々にデクリメントされるべきである。M の 2 5 6 ビットは次の通り分割される。

30

【 1 2 6 2 】

【表 1 2 9 】

M[n]	アクセス	説 明
0	RO	シリアル番号 (16ビット)
1	RO	バッチ番号 (16ビット)
2	MSR	残りシアン (32ビット、ハイ/ロー)
3	NMSR	
4	MSR	残りマゼンタ (32ビット、ハイ/ロー)
5	NMSR	
6	MSR	残りイエロー(32ビット、ハイ/ロー)
7	NMSR	
8 から 11	RO	将来の拡張用 = 0 (64ビット)
12 から 15	RO	ランダムビットデータ (64ビット)

40

ステージ 2 : システム及び消耗品の鍵の決定

どのシステムとどの消耗品が同じ鍵を共有すべきであるかに関する決定がなされると、それらの鍵を定義しなければならない。したがって、 K_1 及び K_2 の値が決定される。殆ど

50

のケースでは、 K_1 及び K_2 は全体で 1 回だけ生成される。連動すべき（現在及び将来の）全てのシステム及び消耗品は、同じ鍵 K_1 及び K_2 の値をもつ。したがって、 K_1 及び K_2 は秘匿されなければならない。なぜならば、システム／消耗品の組み合わせに対するセキュリティメカニズム全体は、鍵が見られてしまうと、役に立たなくなるからである。鍵が暴露されると、その損害は、システム及び消耗品の数、並びに、それらを新たな暴露されていない鍵で再プログラミングすることの容易さに依存する。トナーカートリッジを用いるコピー機の場合、最悪のケースでは、複製品製造者は固有の認証チップを製造し（又は、さらに悪い場合には、それらを購入し）、判明した鍵でチップをプログラムし、固有の消耗品に挿入する。車キー付きの車のケースでは、各車は、異なる鍵の組をもつ。このことにより、2 通りの一般的なシナリオが生まれる。第 1 のシナリオでは、車及び車キーがそれらの鍵を用いてプログラムされた後、 K_1 及び K_2 が削除され、それらの値の記録が保持されず、即ち、 K_1 及び K_2 を暴露する方法がない。しかし、その車のための予備の車キーは、車の認証チップを再プログラミングしない限り製作できない。第 2 のシナリオでは、車製造者は K_1 及び K_2 を保持し、その車のための新しいキーを製作可能である。 K_1 及び K_2 の暴露は、ある者が特定の車専用の車キーを製作できることを意味する。

10

【1263】

したがって、認証チップで使用する鍵及びランダムデータは、非決定論的な手段によって生成されるべきである（完全にコンピュータで生成された疑似乱数は使用できない。なぜならば、それは、生成器のシードを知ることによって将来の全ての数が得られるという点で決定論的であるからである）。 K_1 及び K_2 は、コンピュータではなく、物理的にランダムなプロセスによって生成されるべきである。しかし、ランダム性の自然発生源に基づくランダムビット生成器は、外因の影響を受けやすく、故障し易い。そのような装置は、統計的なランダム性を定期的にテストすべきであることが避けられない。

20

【1264】

簡単ではあるが、有用な乱数発生源は、SGI からの Lavarand（登録商標）システムである。この生成器は、数分間隔で 6 個のラバランプを撮影するデジタルカメラを使用する。ラバランプは無秩序なタービュラント系を含む。得られたデジタル画像は SHA-1 実装部に供給され、SHA-1 実装部は、7 方向のハッシュを生成し、デジタル化された画像から 7 バイト毎に 160 ビット値を生じる。7 組の 160 ビットの合計は 140 バイトになる。この 140 バイトの値が BBS 生成器に供給され、出力ビットストリームのスタート位置を決める。BBS からの出力 160 ビットは、鍵、又は認証チップ 53 である。

30

【1265】

非決定論的なランダムプロセスの極端な例は、クリーンルーム内で、 K_1 のためにコインを 160 回トスし、 K_2 のためにコインを 160 回トスすることである。それぞれの表又は裏に応じて、1 又は 0 が鍵プログラマ装置のパネルに入力される。このプロセスは、（検証のための）数人の観測者が沈黙した状態（誰かが隠しマイクを持っているかもしれない）で行われる。重要なことは、安全なデータ入力及び記憶は、それほど簡単ではないということである。鍵プログラマ装置及び付随するプログラミングステーションの物理的セキュリティは、その固有の完全なドキュメントが必要である。鍵 K_1 及び K_2 が決定された後、それらは、認証チップがその鍵を使用する必要がある限り保持されなければならない。最初の車／車キーのシナリオでは、 K_1 及び K_2 は、単一のシステムチップ及び数個の消耗品チップがプログラムされた後に破壊されている。コピー機／トナーカートリッジのケースでは、 K_1 及び K_2 は、トナーカートリッジがコピー機の役に立っている間に限り保持されなければならない。それらは秘匿されるべきである。

40

【1266】

ステージ 3：システム及び消耗品の MinTicks の決定

MinTicks の値は、認証チップの動作クロック速度（システム固有）と、RD 又は TST 関数呼び出しの間の妥当な時間の根拠（アプリケーション固有）と、に依存する。

50

単一チックの間隔は動作クロック速度に依存する。これは、入力クロック速度と認証チップのクロック制限ハードウェアの最大値である。例えば、認証チップのクロック制限ハードウェアは10MHzにセットされるが（これは変更できない）、入力クロックは1MHzである。このケースでは、1チックの値は10MHzではなく、1MHzが基準になる。入力クロックが1MHzではなく20MHzであるならば、1チックの値は10MHzが基準になる（なぜならば、クロック速度は10MHzで制限されている）。1チックの間隔がわかると、MinTicKs値をセットすることができる。MinTicKsの値は、鍵付きRD又はST関数への呼び出しの間に経過することが要求されるMinTicKs数である。入力クロック速度が10MHzの最大クロック速度と一致する場合を想定する。TSTの呼び出しの間に1秒の最小値を求めるならば、MinTicKsの値は10000000にセットされる。2秒のような値は、（1ページ当たり1回の認証で、1ページが2から3秒毎に生成される）プリンタのようなシステムの場合には全く妥当な値である。

10

【1267】

ステージ4：鍵、ランダムシード、最終チック及び未使用Mのプログラム

認証チップは、製造後、状態が不明である。或いは、認証チップは、ある消耗品で既に使用済みであり、別の消耗品で使用するために再プログラムしなければならない。確認チップの各々は、クリアされ、新しい鍵及び新しい状態データでプログラムされる。認証チップのクリア処理及びその後のプログラミングは、安全なプログラミングステーション環境で行われる。

20

【1268】

信頼されたシステム認証チップのプログラミング

チップが信頼されたシステムチップであるならば、Rのシード値が生成される。それは、物理的にランダムなプロセスから抽出された乱数であり、0ではない。以下のタスクは、以下の順に、安全なプログラミング環境で行われる：

リセット チップをリセット

CLR []

Load R

R（160ビットレジスタ）に物理的ランダムデータをロード

SSI [K₁, K₂, R]

SMT [MinTicKs_{system}]。

30

【1269】

ここで、認証チップはシステムへ挿入するための準備が完了している。認証チップは完全にプログラムされている。システム認証チップがこの時点で盗まれたならば、複製品製造者は、K₁への既知テキスト攻撃に出るために、又はK₂への部分選択テキスト攻撃に出るために、R、F_{K₁}[R]のペアを生成すべくそれらを使用できる。これは、各々が信頼された認証チップを含む多数のシステムの購入と差がない。セキュリティは、認証プロトコルの強度と、K₁及びK₂のランダム性と、に依存する。

【1270】

信頼されない消耗品認証チップのプログラミング

チップが信頼されない消耗品認証チップである場合、プログラミングは信頼されたシステム認証チップのプログラミングとは少し異なる。最初に、Rのシード値は0である。それは、M及びAccessModeの値用の付加的なプログラミングを行う。未使用M[n]は0でプログラムされ、ランダムM[n]はランダムデータでプログラムされる。以下のタスクは、以下の順に、安全なプログラミング環境で行われる：

リセット チップをリセット

CLR []

Load R

R（160ビットレジスタ）に0をロード

SSI [K₁, K₂, R]

40

50

L o a d X

X (256ビットレジスタ) に0をロード

S e t 適当なM [n] に対応したXに物理的ランダムデータをセット

W R [X]

L o a d Y

Y (32ビットレジスタ) に0をロード

S e t 適当なM [n] に対応したYにリードオンリーアクセスモードをセット

S A M [Y]

S M T [M i n T i c k s C o n s u m a b l e] 。

【1271】

ここで、信頼されない消耗品チップは一般的な状態データでプログラムする準備が完了している。認証チップがこの時点で盗まれたならば、攻撃者は、制限選択テキスト攻撃を実行できる。最もよい状況では、Mの一部分はリードオンリー（0及びランダムデータ）であり、Mの残りは攻撃者によって（WRコマンドを用いて）完全に選択される。攻撃者による多数のRD呼び出しは、限定されたMに対する $F_{K2}[M|R]$ を獲得する。最悪の状況では、Mは攻撃者によって完全に選択される（なぜならば、256ビットの全てが状態データに使用されるためである）。しかし、どちらの場合でも、攻撃者はRの値を選択できない。なぜならば、RはRNDへの呼び出しによってシステム認証チップから供給されるためである。選択されたRを獲得する唯一の方法は、ブルートフォース攻撃である。ステージ4及び5が同じプログラミングステーションで実行された場合（好ましく、且つ理想的な状況）、認証チップはステージ間で取り外せないことに注意すべきである。この場合、認証チップがこの時点で盗まれる可能性はない。認証チップを1回でプログラムするか、又は2回でプログラムするかの決定は、システム／消耗品製造者の要求次第である。

【1272】

ステージ5：状態データ及びアクセスモードのプログラム

このステージは、消耗品認証チップの場合だけに必要である。なぜならば、M及びAccess Modeレジスタはシステム認証チップ上では変更できないからである。M [n]の未使用及びランダム値は既にステージ4でプログラムされている。残りの状態データをプログラムする必要がある、関連したAccess Mode値をセットする。このステージの速度は、MinTicksレジスタに格納された値によって制限されることに注意する必要がある。このステージは、ステージ4とステージ5が実行される場所／時間の物理的場所の違い又は時間の違いを考慮して、ステージ4から分離されている。理想的には、ステージ4及び5は同時に同じプログラミングステーションで実行される。ステージ4は有効認証チップを生成するが、それらに初期状態値（0以外）をロードしない。これは、チップのプログラミングを消耗品の生産ラインの運行と一致させることができる。ステージ5は何回も実行し、毎回、異なる状態データ値及びアクセスモード値を設定することが可能であるが、1回だけ実行され、残りの全ての状態データ値をセットし、残りの全てのアクセスモード値をセットする可能性の方が高い。例えば、生産ラインがセットアップされ、認証チップのバッチ番号及びシリアル番号が生産される物理的な消耗品に応じて生成される。これを一致させることは、状態データが物理的に異なる工場でロードされる場合にはかなり難しくなる。

【1273】

ステージ5のプロセスは、最初に、チップが有効消耗品チップであることを保証するために検査を行い、ここでは、認証チップからデータを収集するRDが含まれ、次に、初期データ値のWRが行われ、更に、新しいデータ値を永久的にセットするためSAMが行われる。これらのステップの概要を次に記載する：

I s T r u s t e d = G I T [] ；

I s T r u s t e dがセットされているならば、エラー（間違った種類のチップ！）で終了する；

10

20

30

40

50

有効な入力ペアを取得するため有効システムチップの R N D を呼び出し；
 有効な入力ペアを渡して、プログラムされるべきチップの R D を呼び出し；
 認証チップの R D からの結果を X（256 ビットレジスタ）にロードする；
 X 及び消耗品チップが有効であることを保証するため有効システムチップの T S T を呼び出し；
 T S T が 0 を返したならば、エラー（システムに対し間違った消耗品チップ）で終了する。

X のビットに初期状態値をセット；

W R [X]；

Y（32 ビットレジスタ）に 0 をロード；

新しい状態値のためのアクセスモードに対応した Y のビットをセット；

S A M [Y]。

【1274】

勿論、検証（ステージ 1 から 7）は、ステージ 4 及び 5 が同じプログラミングステーションで次々に行われる場合には行わなくてもよい。しかし、この検証は、ステージ 5 がステージ 4 とは別個のプログラミングプロセスとして実行される他の全ての状況では行われるべきである。これらの認証チップがここで盗まれた場合には、それらは既に特定の消耗品で使用するためにプログラムされている。攻撃者は、盗んだチップを複製消耗品に取り付けることができる。このような盗難は、複製された製品の個数を盗難されたチップの個数に制限する。1 回の盗難では、複製品製造者が費用対効果に優れたビジネスを行うために十分な供給数が得られない。このチップの別の用途は、攻撃者が部分選択テキスト攻撃又はブルートフォース攻撃に出るために、認証チップ付きの同数の消耗品を購入しなくても済むことである。このような攻撃が行われても、鍵の特別な安全性の欠陥はない。

【1275】

製造

認証チップの回路は物理的攻撃に対抗できなければならない。製造インプリメンテーションガイドラインの概要を説明し、その後、チップの物理的防御の仕様を（攻撃の順に）説明する。

【1276】

製造のガイドライン

製造に関する認証チップの実装用の一般的なガイドラインを以下に列挙する：

標準プロセス

最小サイズ（なるべく）

クロックフィルタ

雑音発生器

タンパー防止及び検出回路

タンパー検出付き保護メモリ

プログラムコードのローディング用ブート回路

鍵データパス用の F E T の特殊実装

可能な場所でのポリシリコン層内のデータコネクション

過小過多電力検出ユニット

テスト回路無し。

【1277】

標準プロセス

認証チップは（フラッシュのような）標準的な製造プロセスで実装できる。これは、製造場所の非常に広範囲の選択を可能にさせること

明確であり、且つ正常に動作するテクノロジーを使用すること

コストを低減すること

を充たすために必要である。標準プロセスは物理的な保護メカニズムを可能にさせることに注意する必要がある。

10

20

30

40

50

【1278】

最小サイズ

認証チップ53は、低コスト消耗品のための認証メカニズムとして組み込むことができるように製造コストを低くしなければならない。したがって、チップサイズはできるだけ小さいままにすることが望ましい。各認証チップは、802ビットの不揮発性メモリが必要である。その上、最適化されたHMAC-SHA1に必要な記憶容量は1024ビットである。チップの残りの部分（状態機械、プロセッサ、CPU、又はプロトコル3を実施するために選択されたもの）は、トランジスタの個数が最小限に抑えられ、チップ当たりのコストが最小限になるように最低限に維持される。秘密鍵情報を保有する回路面積、また垂直h、鍵に関する情報が漏れる虞のある回路面積も最小限に抑えられるべきである（特殊データパスのための以下のノンフラッシングCMOSを参照せよ）。

10

【1279】

クロックフィルタ

認証チップ回路は特定のクロック速度レンジで動作するように設計される。ユーザはクロック信号を直接供給するので、攻撃者は、処理中の特定の時点で回路に乱調状態を生じさせる可能性がある。この一例は、高いクロック速度（回路が設計対象とした速度よりも高い）は、XORが適切に動作することを妨げ、2個の入力のうちの最初の入力があるまま返される場合である。このようなスタイルの過渡現象障害攻撃は、秘密鍵情報を復元する際に非常に有効である。ここから学ぶべきことは、入力クロック信号は信頼できないということである。入力クロック信号は信頼できないので、最大周波数までで動作するように制限されるべきである。これはいろいろな方法で実現できる。クロック信号をフィルタ処理する一つの方法は、「エッジオン」を遅延器へ渡すエッジ検出ユニットを使用することであり、次に、遅延によって入力クロック信号の通過を許可する。図174は、クロックフィルタ内のクロック信号フローの説明図である。遅延は、最大クロック速度が特定の周波数（例えば、4MHz）になるようにセットされる。この遅延はプログラマブルではなく、固定であることに注意すべきである。フィルタ処理されたクロック信号は、必要に応じて内部で更に分周される。

20

【1280】

雑音発生器

各認証チップは連続した回路雑音を発生する雑音発生器を含む。雑音は、チップの通常の動作からの他の電磁放射と干渉し、雑音をI_{dd}信号に加える。雑音発生器の設置は、放射波長の長さのために認証チップに問題を起こさない。雑音発生器は、電子雑音、クロックサイクル毎の複数状態変化を発生し、タンパー防止及び検出回路用の疑似ランダムビットソースとして使用される。雑音発生器の簡単な実装は、非零の数でシードされた64ビットLFSRである。雑音発生器で使用されるクロックは、できるだけ多くの雑音を発生させるため、チップの最大クロックレートで動く。

30

【1281】

タンパー防止及び検出回路

1組の回路が認証チップへの物理的攻撃のテスト及び防止のために必要である。しかし、実際に攻撃として検出される者は、意図的な物理的攻撃ではないかもしれない。したがって、認証チップ内で以下の2タイプの攻撃、即ち、

40

物理的攻撃の出現を確信できる場合

物理的攻撃の出現を確信できない場合

を区別することが重要である。

【1282】

この2タイプの検出は、検出の結果として実行されることが違う。第1のケース、即ち、物理的攻撃の出現を確信できる場合、フラッシュメモリ鍵情報の消去が目的になった動作である。第2のケース、即ち、物理的攻撃の出現を確信できない場合、何かの間違ひがあるかもしれない。動作を行うべきではあるが、秘密鍵情報を消去する動作ではない。第2のケースでとるべき適切な動作は、チップリセットである。検出されたものがチップに

50

永久的な損傷を加える攻撃であるならば、次にも同じ状態が出現し、チップは再度リセットする。これに対して、検出されたものがチップの通常の動作環境の一部であるならば、このリセットは鍵を損ねない。

【1283】

回路が状況を理解できない事象の良い例は、電源誤動作である。この誤動作は、鍵に関する情報を暴こうとする意図的な攻撃かもしれない。しかし、それは、接続間違い、又は単に、電源遮断シーケンスの開始かもしれない。したがって、チップをリセットだけすることが最良であり、鍵を消去しない。チップが電源遮断中であれば、何も失われない。システムが故障している場合、リセットの繰り返しによって、消費者はシステムを修理する。両方のケースで、消耗品は損なわれていない。回路が状況を理解できる事象の良い例は、チップ内のデータラインの切断である。この攻撃が何らかの方法で検出された場合、これは、チップの故障（製造欠陥）、又は攻撃の結果に限られる。何れのケースでも、秘密情報の消去は、目的にかなった選択すべきステップである。

10

【1284】

この結果として、各認証チップは図示されるように2本のタンパー検出ラインを含む。一方は明確な攻撃用であり、もう一方は攻撃可能性のためのものである。これらのタンパー検出ラインには、多数のタンパー検出テストユニットが接続され、各ユニットは様々な形態のタンパーリングをテストする。その上、タンパー検出ライン及び回路自体は不正に手を加えられないことを保証したい。

【1285】

タンパー検出ラインの一方の端は、（一般的な動作回路よりも高速でクロッキングする）疑似ランダムビットのソースである。上記の雑音発生器回路は適当なソースである。生成されたビットは、2種類のパスを通り、一方のパスは原データを搬送し、他方は原データの反転を搬送する。これらのビットを搬送する配線は、一般的なチップ回路（例えば、メモリ、鍵操作回路等）よりも上の層にある。配線はランダムビット発生器もカバーする。ビットは、XORゲートによって多数の場所で再結合される。ビットが異なる場合（そうあるべきである）、1が出力され、特定のユニットによって使用される（例えば、メモリ読み出しからの各出力ビットはこのビット値とAND演算される）。ラインは、最終的に、フラッシュメモリ消去回路に集まり、そこで、XORからの0によって完全な消去がトリガーされる。ラインには多数のトリガーが取り付けられ、各々はチップ上の物理的攻撃を検出する。各トリガーは、GNDへ繋がれた大型nMOSトランジスタを有する。タンパー検出ラインは、このnMOSトランジスタを物理的に通る。テストが失敗したとき、トリガーはタンパー検出ラインを0にする。XORテストは、このクロックサイクル、又は次のクロックサイクル（平均的に）の何れかで失敗し、チップをリセット又は消去する。図175は、テストと、消去回路又はリセット回路の何れかに接続されたXORと、を用いるタンパー検出回路の基本原理の説明図である。

20

30

【1286】

タンパー検出ラインは、図176の大型nMOSトランジスタのレイアウトに示されているように、各Testの出力トランジスタのドレインを通る。タンパー検出ラインは切断できない。なぜならば、これは、ランダムソースからの1と0の流れを止めるからである。それにより、XORテストは失敗する。タンパー検出ラインは、各テストを物理的に通るので、タンパー検出ラインを切断することなく、特定のテストを除去することは不可能である。重要なことは、XORは、攻撃の機会を減らすため、タンパー検出ラインに沿って様々な場所からの値を取ることである。図177は、チップの様々な部分で使用されるタンパー検出ラインから多数のXORをとる状況を示している。これらのXORの各々は、各ユニット又はサブユニットで使用されるチップ正常ビットChipOKを生成しているとみなすことができる。

40

【1287】

サンプル用途としては、各ユニットにOKビットを設け、これを各サイクルで与えられたChipOKビットとAND演算することがある。OKビットはリセットで1がロードさ

50

れる。OKビットが0であるならば、ユニットは次のリセットまで失敗（フェイル）である。タンパー検出ラインが正しく機能するならば、チップは、リセットを行うか、又は全ての鍵情報を消去する。リセット回路又は消去回路が破壊されているならば、このユニットは機能しないので、攻撃者を阻止する。リセットライン及び消去ライン、並びに、関連した回路の行き先は、非常に状況に敏感である。それは、個別のタンパーテストと非常に類似した方法で保護されるべきである。攻撃者がリセット回路に繋がる配線を簡単に切断できるのであれば、リセットパルスを生成しても意味がない。実際の実装形態は、リセットでクリアされるべき項目と、それらの項目をクリアする方法と、に依存する部分が多い。

【1288】

最後に、図178は、タンパーラインがチップの雑音発生器回路を保護する様子を示す図である。発生器及びNOTゲートは同じ高さにあり、一方、タンパー検出ラインは発生器よりも高いところを通る。

【1289】

タンパー検出付き保護メモリ

秘密情報又はプログラムコードはフラッシュメモリに記憶するだけでは不十分である。フラッシュメモリ及びRAMは、プログラムコード又は鍵情報の特定のビットを変更（又はセット）使用とする攻撃者から保護されなければならない。使用されるメカニズムは、（上述の）タンパー検出回路で使用されているメカニズムに準拠する。ソリューションの第1の部分は、タンパー検出ラインが各フラッシュ又はRAMビットの直ぐ上を確実に通ることである。これは、攻撃者がフラッシュ又はRAMの内容をプローブできないことを保証する。配線の保護部の破損はタンパー検出ラインの切断である。この破損は、消去信号をセットするので、メモリの内容が消去される。タンパー検出ライン上の高周波雑音も受動的な観測の妨げになる。

【1290】

フラッシュに対するソリューションの第2の部分は、多重レベルデータ記憶装置を使用し、しかも、それらの多重レベルのうちのサブセットだけを有効ビット表現のために使用することである。一般的に、多重レベルフラッシュ記憶装置が使用されるとき、単一のフローティングゲートが2ビット以上を保持する。例えば、4電圧状態トランジスタは、2ビットを表現できる。最小電圧及び最大電圧がそれぞれ00及び11で表すことにすると、2個の中間電圧は01及び10を表す。認証チップの場合、単一ビットを表現するために2個の中間電圧を使用し、両端の2個は無効状態であるとみなす。攻撃者がゲートの回路を閉じる、又は切断することにより、ビットの状態を強制的にある方向又は別の方向にさせようとした場合に、無効電圧（従って、無効状態）が現れる。

【1291】

RAMに対するソリューションの第2の部分は、パリティビットを使用することである。レジスタのデータ部分は（攻撃後には一致しなくなる）パリティビットを使って検査できる。フラッシュ及びRAMから到来するビットは、共通タンパー検出ラインに接続された（1ビットに一つずつの）多数のテストユニットによって確認される。タンパー検出回路は、データが通過する最初の回路である（したがって、攻撃者によるデータラインの切断を阻止する）。

【1292】

プログラムコードのローディング用ブート回路

プログラムコードは、ROMではなく、多重レベルフラッシュに保持されるべきである。なぜならば、ROMはテストできない方法で変更されやすいからである。したがって、ブートメカニズムは、プログラムをフラッシュメモリに取り込むために必要である（フラッシュメモリは製造後中間状態になっている）。ブート回路はROM内には置かれず、小型の状態機械で十分である。そうでなければ、ブートコードは検出できない方法によって変更される。ブート回路は、全てのフラッシュメモリを消去し、消去が正しく動作したことを確認し、次に、プログラムコードをロードする。フラッシュメモリは、プログラムをロ

10

20

30

40

50

ーディングする前に消去しなければならない。そうでなければ、攻撃者はチップをブート状態に置き、既存の鍵を抽出しただけのプログラムをロードすることができる。状態機械は、新しいプログラムコードをロードする前に、全てのフラッシュメモリがクリアされていることを保証するため（攻撃者が消去ラインを切断していないことを保証するため）検査を行う。プログラムコードのローディングは、（鍵のような）秘密情報がロードされる前に、安全なプログラミングステーションによって行われる。

【1293】

鍵データバス用の特殊な FET の実装

CMOS インバータ（nMOS トランジスタと組み合わされた pMOS トランジスタを含む）場合の通常の FET 実装の状況が図 179 に示されている。遷移中に、nMOS トランジスタと pMOS トランジスタの両方が中間的な抵抗値をもつ短い期間が存在する。その結果としての電源－グラウンド短絡回路は、電流を一時的に増大させ、實際上、CMOS 装置によって消費される電流の大部分を占める。少量の赤外線光が短絡回路の期間に放出され、シリコン基板を介して観測することができる（シリコンは赤外線を透過する）。また、少量の光がトランジスタゲート容量と伝送ライン容量の充電中及び放電中に放出される。

【1294】

秘密鍵情報を操作する回路の場合、このような情報は秘匿されなければならない。代替的なノンフラッシング CMOS 実装が、鍵、又はその鍵に基づいて部分的に計算された値を操作する全てのデータバスのため使用される。二つの重なり合わないクロック $\phi 1$ 及び $\phi 2$ の使用によって、フラッシュしないメカニズムが得られる。 $\phi 1$ は全ての nMOS トランジスタの第 2 ゲートに接続され、 $\phi 2$ は全ての pMOS トランジスタの第 2 ゲートに接続される。遷移は、クロックと組み合わされた場合に発生する。 $\phi 1$ 及び $\phi 2$ は重なり合わないで、pMOS 及び nMOS トランジスタは、同時に中間的な抵抗値をとることがない。その構成は図 180 に示されている。

【1295】

最後に、通常の CMOS インバータは、重要なノンフラッシング CMOS コンポーネントの近傍に配置することができる。これらのインバータは上記のタンパー検出ラインからそれらの入力信号を取り出す。タンパー検出ラインは通常の動作回路の何倍も高速に動作するので、正味の影響は、ノンフラッシング CMOS コンポーネントの各々の場所付近に高いレートで生じる光バーストである。明るい光は、付近の弱い光の観測を邪魔するので、観測者は、チップそのものでどのようなスイッチング動作が行われているかを検出し得ない。これらの通常の CMOS インバータは、回路雑音の量を効果的に増加させ、SN 比を低下させ、役に立つ EMI を覆い隠す。

【1296】

ノンフラッシング CMOS の使用によって多数の副作用が現れる：

チップの実効速度は、1 クロックサイクル当たりのクロック立ち上がり時間の 2 倍だけ減少する。これは、認証チップの場合、問題にならない；

ノンフラッシング CMOS によって引き出される電流量が減少する（なぜならば、短絡回路が現れないからである）。しかし、これは、通常の CMOS インバータの使用によって相殺される；

クロックの配線によってチップ面積が増加し、特に、 $\phi 1$ 及び $\phi 2$ の多数のバージョンが様々な伝搬のレベルを考慮に入れるために要求されるからである。チップ面積の推定量は通常の実装の 2 倍である；

認証チップのフラッシュしないエリアの設計は、通常の CMOS 設計を用いた場合よりも多少複雑になる。特に、標準的なセルコンポーネントが使用できず、そのエリアがフルカスタム化されるからである。これは、認証チップのように小さいものの場合、特に、チップ全体をこの方法で保護しなくても構わないときには、問題にならない。

【1297】

可能な場所でのポリシリコン層内の配線

10

20

30

40

50

どこでも可能であれば、鍵又は秘密データが流れる配線は、ポリシリコン層内に作られるべきである。必要であれば、メタル1に設けられるが、決して（タンパー検出ラインを含む）最上部のメタル層に設けるべきではない。

【1298】

過小過多電力検出ユニット

各認証チップは、電源攻撃を妨げるため、過小過多電力検出ユニットが必要である。過小過多電力検出ユニットは、電源誤動作を検出し、電源レベルがある許容範囲に収まることを保証するため、電圧基準に対して電源レベルをテストする。このユニットは、単一の電圧基準と、2個のコンパレータを含む。過小過多電力検出ユニットは、リセットタンパー検出ラインに接続してもよく、これにより、トリガーされたときにリセットを引き起こす。過小過多電力検出ユニットの副作用は、電源遮断中に電圧が降下したとき、リセットがトリガーされ、作業レジスタを消去することである。

10

【1299】

テスト回路無し

認証チップ上のテストハードウェアは簡単に脆弱性を持ち込み得る。その結果として、認証チップは、BIST又はスキャンパスを含むべきではない。したがって、認証チップは外部テストベクトルを用いてテスト可能にされる。これは、認証チップが複雑ではないので実現可能である。

【1300】

ROM読み出し

この攻撃は、鍵がアドレス指定可能なROMに記憶されていることに依存する。各認証チップはその認証鍵を、アドレス指定可能なROMではなく内部フラッシュメモリに記憶しているので、この攻撃は無関係である。

20

【1301】

チップのリバースエンジニアリング

チップのリバースエンジニアリングは、認証の安全性がアルゴリズムに由来している場合に限り有用である。しかし、本発明の認証チップは、アルゴリズムの秘密性ではなく、秘密鍵に依存している。これに対して、本発明の認証アルゴリズムは公開されていて、いずれにしても、大量生産消耗品の攻撃者は、チップの内部の詳細な設計図を獲得する可能性があったと考えられる。これらの要因から見て、チップ自体のリバースエンジニアリングは、記憶されたデータとは対照的に、脅威ではない。

30

【1302】

認証プロセスの不当使用

この攻撃がとり得る形態は幾つかあり、各々の形態は成功の程度に違いがある。全てのケースにおいて、複製品製造者はシステム設計と消耗品設計の両方を入手できると考えられる。ある攻撃者は、認証コードを生成するのではなく、システムを騙して有効なコードを返すチップを構築しようとする。この攻撃は以下の二つの理由から実現不可能である。第1の理由は、システム認証チップ及び消耗品認証チップは、物理的には同じであるとしても、別々にプログラムされている点にある。特に、RDオペコード及びRNDオペコードは同一であり、WRオペコードとTSTオペコードも同様である。システム認証チップはRDコマンドを実行できない。なぜならば、全ての呼び出しは、RDではなく、RNDへの呼び出しとして解釈されるからである。この攻撃が失敗する第2の理由は、別々のシリアルデータラインがシステムからシステム認証チップ及び消耗品認証チップへ設けられるからである。その結果として、どちらのチップも、もう一方へ送信される内容、或いは、もう一方から受信される内容を見ることができないからである。攻撃者が、（消耗品残量をデクリメントする）WRコマンドを無視する複製チップを構築した場合、プロトコル3は、その後のRDがWRは出現しなかったことを検出することを保証する。システムは、したがって、消耗品を使用しなくなり、攻撃者を阻止する。これは、攻撃者が認証前に接触の欠損をシミュレートした場合にも成り立ち、認証が行われないので、消耗品の使用は発生しない。したがって、攻撃者は、複製消耗品が受け付けられるように各システムを変

40

50

更する範囲に制限される。

【1303】

システムの変更

変更の最も簡単な方法は、システムの認証チップを、T S Tへの呼び出し毎に成功だけを通知するチップと置き換えることである。これは、認証毎に何回もT S Tを呼び出し、最初の数回は偽の値を与え、T S Tからの失敗の通知を要求するシステムによって妨げられる。T S Tへの最後の呼び出しは成功することが予想される。T S Tの偽の呼び出しの回数は、R D、又はシステムクロックから戻された結果の一部分から決定してもよい。残念ながら、攻撃者は、単に、システムを書き換えることができるので、新しいシステム複製認証チップ53は、消耗品チップ又はクロックから返された結果を監視できる。複製システム認証チップは、監視された値がそのT S T関数へ与えられたときに、成功しか返さない。複製消耗品はR Dのハッシュ結果として任意の値を返すことができ、複製システムチップはその値を有効であることを明らかにする。したがって、システムがシステム認証チップを何回も呼び出して意味がない。なぜならば、書き換え攻撃は、全てのシステムに対してではなく、再配線されたシステムだけに対して正しく動作するからである。システムに対する類似した攻撃の形態は、システムROMの取り替えである。ROMプログラムコードは、認証が行われることの無いように変更可能である。これに関しては何もできない。なぜならば、システムは消費者の手にあるからである。勿論、これはあらゆる保証を無効にするが、消費者は、複製消耗品が非常に安価であり、且つ本物の品物よりも簡単に入手できるのであれば、変更する価値があると考えらるであろう。

10

20

【1304】

システム／消耗品製造者は、したがって、この種類の攻撃がどの程度起こり得るかを判断しなければならない。このような調査は、システム及び消耗品の価格構成、システムサービスの頻度、物理的な変更を加えたときの消費者の利益、消費者がその変更を実行するために出向く場所を当然に含む。システムを変更する限界のケースは、複製品製造者が、複製消耗品を必要とする完全な複製システムを提供する場合である。これは、単なる競合、又は特許権の侵害である。いずれにしても、これは認証チップの目的の範囲外であり、複製される技術やサービスに依存する。

【1305】

従来のプロービングによるチップ動作の直接視

チップ動作を視察するためには、チップは動作していなければならない。しかし、タンパー防止及び検出回路は、鍵を処理又は保持するチップの区域を覆う。これらの区域を、タンパー防止ラインを通して見ることは不可能である。攻撃者は、単にタンパー防止層を通してチップをスライスできない。なぜならば、そうすることによって、タンパー検出ラインが破断し、電源投入時に全ての鍵を消去するからである。単に消去回路を破壊するだけでは不十分である。なぜならば、認証チップ内の多数のユニットに供給される多数のChip OKビット（このとき全て0）は、チップの通常の動作回路の機能を停止させるからである。攻撃のためにチップをセットアップするためには、攻撃者は、タンパー検出ラインを削除し、フラッシュメモリの消去を停止し、Chip OKラインを信頼していたコンポーネントを何とかして書き換えることが必要である。たとえ、この全てを実行できたとしても、チップをこのレベルまでスライスする動作は、鍵を保持する不揮発性メモリ内の電荷パターンを殆ど破壊し、プロセスを無益にするであろう。

30

40

【1306】

不揮発性メモリの直接視

認証チップがフラッシュメモリのフローティングゲートを露出させるようにスライスされ、それらを放電させない場合、鍵は、おそらくSTM又はSKMを用いて直接視できるであろう。しかし、ゲートを放電させずにチップをこのレベルまでスライスすることは、おそらく不可能である。湿式エッチング、プラズマエッチング、イオンミリング、又は化学機械研磨は、ほぼ確実にフローティングゲート上に存在する小さい電荷を放出する。これは、通常のフラッシュメモリの場合に成り立つが、多重レベルフラッシュメモリの場合に

50

はより確かである。

【1307】

状態変化によって生じる光バーストの監視

秘密鍵情報を操作する回路の全ての区域は、上記のノンフラッシングCMOSに実現される。これは、大部分の光バーストの放出を阻止する。ノンフラッシングCMOSの直ぐ近くに配置された通常のCMOSインバータは、キャパシタの充電及び放電によって生じる僅かな放射を見えなくする。インバータはタンパー検出回路に接続されるので、インバータは、ノンフラッシングCMOSの状態変化毎に何回も（高いクロックレートで）状態を変化する。

【1308】

EMIの監視

上記の雑音発生器は回路雑音を誘起する。雑音は、チップの通常の動作からの他の電磁放射を妨害し、内部データ転送の有意な読み出しが曖昧になる。

【1309】

I_{dd}変動の監視

この種の攻撃に対するソリューションは、I_{dd}信号におけるSN比を減少させることである。これは、回路の雑音量を増大させ、信号量を減少させることにより実現される。雑音発生器回路（これは、EMI攻撃に対する防御としても機能する）は、I_{dd}中の重要な情報を目立たなくするため、サイクル毎に十分な状態変化を引き起こす。その上、チップの鍵を伝搬するデータパスの特別なノンフラッシングCMOS実装は、状態変化が生じたときに電流が流れることを阻止する。これは、信号量を減少させる利点がある。

【1310】

差分故障解析

差分故障ビット誤りは、イオン化、マイクロ波放射、又は環境ストレスによって、目標とした形ではなく、取り込まれる。この種の攻撃の最も起こりやすい影響は、フラッシュメモリの変更（無効な状態を引き起こす）又はRAMの変更（不良パリティ）である。無効な状態及び不良なパリティは、タンパー検出回路によって検出され、鍵の消去を生じさせる。タンパー検出ラインは鍵操作回路を覆うので、鍵操作回路に取り込まれたあらゆる誤りは、タンパー検出ラインの誤りによって反映される。タンパー検出ラインが影響されるならば、チップは、継続的にリセットをするか、又は単にパワーアップ時に鍵を消去し、攻撃を無益にする。目標としていない攻撃に頼り、「チップのちょうど良い部分がちょうど良い方法で影響を受ける」ことを望むのではなく、攻撃者は、（上書き攻撃、ゲート破壊等の）目標とした故障を持ち込むことをやってみる方がよい。

【1311】

クロック誤作動攻撃

（上記の）クロックフィルタはクロック誤動作攻撃の実現性を除外する。

【1312】

電源攻撃

（上記の）過小過大電力検出ユニットは電源攻撃の実現性を除外する。

【1313】

ROMの上書き

認証チップは、プログラムコード、鍵、及び秘密情報を、ROMではなく、フラッシュメモリに格納する。したがって、この攻撃は可能性がない。

【1314】

EEPROM／フラッシュの変更

認証チップは、プログラムコード、鍵、及び秘密情報をフラッシュメモリに格納する。しかし、フラッシュメモリは、2本のタンパー防止及び検出ラインによって覆われている。これらのラインの何れか一方が（ゲートを破壊するプロセスにおいて）破断されると、攻撃はパワーアップ時に検出され、各チップは、（連続的に）リセットするか、又はフラッシュメモリから鍵を消去する。しかし、たとえ、攻撃者が何とかしてフラッシュメモリの

10

20

30

40

50

ビットへアクセスし、特定のビットを保持しているゲートを破壊するか、若しくは、省くことができるとしても、これは、そのビットを強制的に無充電、又はフル充電の状態にする。これらはどちらも、認証チップの多重レベルフラッシュメモリの使用法では無効状態である（二つの中間状態だけが有効である）。データ値がフラッシュから転送されたとき、検出回路は、消去タンパー検出ラインをトリガーさせ、これにより、フラッシュメモリの残りの部分を消去し、チップをリセットする。したがって、EEPROM／フラッシュメモリの変更攻撃は無益である。

【1315】

ゲート破壊攻撃

ゲート破壊攻撃は、単一ゲートを変更し、動作中にチップに情報を暴露させる攻撃者の能力に依存している。しかし、秘密情報を操作する回路は、2本のタンパー防止及び検出ラインの一方によって覆われている。これらのラインの何れか一方が（ゲートを破壊するプロセスにおいて）破断されると、攻撃はパワーアップ時に検出され、各チップは、（連続的に）リセットするか、又はフラッシュメモリから鍵を消去する。この種の攻撃に出るためには、攻撃者は、最初に、どのゲートを目標にすべきかを決定するためにチップをリバースエンジニアリングしなければならない。目標ゲートの場所が決定された後、攻撃者は、保護しているタンパー検出ラインを破断し、フラッシュメモリの消去を停止し、Chip OKラインを信頼していたコンポーネントを何とかして書き換えることが必要である。回路の書き換えはチップをスライスすることなく実行できず、たとえ、実行できたとしても、チップをこのレベルまでスライスする動作は、鍵を保持する不揮発性メモリ内の電荷パターンを殆ど破壊し、プロセスを無益にするであろう。

【1316】

上書き攻撃

上書き攻撃は、鍵の個々のビットを、その前の値を知ることなく、セットし得ることに依存している。それは、従来のプロービング攻撃と同じようにチップをプロービングし、ゲート破壊攻撃と同じようにゲートを破壊することに依存する。これらの両方の攻撃は（それぞれのセクションで説明したように）タンパー防止及び検出回路とChip OKラインを使用しているので成功しない。しかし、たとえ、攻撃者が何とかしてフラッシュメモリのビットへアクセスし、特定のビットを保持しているゲートを破壊するか、若しくは、省くことができるとしても、これは、そのビットを強制的に無充電、又はフル充電の状態にする。これらはどちらも、認証チップの多重レベルフラッシュメモリの使用法では無効状態である（二つの中間状態だけが有効である）。データ値がフラッシュから転送されたとき、検出回路は、消去タンパー検出ラインをトリガーさせ、これにより、フラッシュメモリの残りの部分を消去し、チップをリセットする。同じように、RAMから読み出された改ざんされている値のパリティチェックによって、消去タンパー検出ラインがトリガーされる。したがって、上書き攻撃は無益である。

【1317】

残留メモリ攻撃

認証チップ内の作業用レジスタ又はRAMは、電源が取り外されたとき、認証鍵の一部を保持している。作業用レジスタ及びRAMは、電源の取り外し後、ある時間その情報を保持し続ける。チップがスライスされ、レジスタ／RAMのゲートが、それらを放電させることなく露出されたとき、データは、STMを使用して、おそらく直接視できるであろう。第1の防御は、上述の電源誤動作攻撃に対する防御の説明からわかる。電源が取り外されると、リセット状態がメモリのクリアを引き起こすのと同じように、全てのレジスタ及びRAMがクリアされる。この攻撃が成功する見込みは、フラッシュメモリの読み出しの見込みよりもひく。RAM電荷は（本質的に）フラッシュメモリよりも簡単に失われる。RAMを明らかにするためのチップのスライス化は、（電荷が、単に、メモリがリフレッシュされていないこと、及びスライス化を実行するために要する時間を原因として失われていないとしても）確実に電荷を失わせる。したがって、この攻撃は無益である。

10

20

30

40

50

【1318】

チップ窃盗攻撃

認証チップの耐用期間には、区別可能なフェーズが存在する。チップは、以下のステージの何れかのときに盗まれる可能性がある：

製造後、鍵をプログラミングする前；

鍵のプログラミング後、状態データをプログラミングする前；

状態データのプログラミング後、消耗品又はシステムに挿入する前；

システム又は消耗品に挿入された後。

【1319】

チップ製造とプログラミングステーションの間の盗難は、複製品製造者にブランクチップを与えるだけである。これは、認証チップの販売を危うくするだけであり、認証チップによって認証されるものはない。プログラミングステーションは、消耗品及びシステム製品鍵をもつ唯一のメカニズムであり、複製品製造者はチップを正しい鍵でプログラミングできない。複製品製造者は、自分専用のシステム及び消耗品用にブランクチップをプログラミングすることは可能であるが、それらの品を見破られずに販売することは困難であろう。盗難の第2の形態は、認証チップが2以上の別個のプログラミングフェーズを通るような状況だけで起こり得る。これは起こり得るが、可能性は低い。いずれにしても、最悪の状況は、状態データがプログラミングされておらず、全てのMがリード／ライトになっている場合である。このケースでは、攻撃者はチップに対して適応的選択テキスト攻撃に出ることが可能である。HMAC-SHA1アルゴリズムはこの攻撃に対抗できる。盗難の第3の形態は、プログラミングステーションと取付工場との間で行われる。認証チップは、既に特定のシステム用、又は特定の消耗品用にプログラムされている。これらのチップの唯一の用途として、泥棒はこれらを複製システム又は複製消耗品に収容する。複製システムは見当違いであり、複製システムは認証チップ53を必要とさえしないであろう。複製消耗品の場合、このような盗難は、複製品の数を盗み取られたチップの数に制限する。1回の窃盗は、複製品製造者が費用対効果に優れたビジネスを行うために十分な供給数が得られない。盗難の最後の形態は、システム又は消耗品自体の盗難である。盗難が製造者側で発生した場合、物理的セキュリティプロトコルを強化しなければならない。盗難が他の場所で行われた場合、それは、その品の所有者、及び警察又は保険会社だけの問題である。認証チップが使用するセキュリティメカニズムは、消耗品及びシステムが一般の人の支配下にあることを前提としている。その結果として、それらが盗まれても、鍵の安全性に重大な影響はない。

【1320】

認証チップ設計

認証チップは、物理的及び論理的的外部インタフェースをもつ。物理的インタフェースは認証チップをどのように物理的システムに接続できるかを規定し、論理的インタフェースはシステムがどのように認証チップと通信できるかを規定する。

【1321】

物理的インタフェース

認証チップは、小型の4ピンCMOSパッケージである（実際の内部サイズは、 $0.25\text{ }\mu\text{m}$ フラッシュプロセスを使用する約 0.30 mm^2 である）。4ピンは、グラウンド（GND）、クロック（CLK）、パワー（Power）、及びデータ（Data）である。パワーは公称電圧である。電圧が公称電圧からある量以上変動すると、チップはリセットする。推奨されるクロック速度は4から10MHzである。内部回路は、クロック信号をフィルタ処理し、安全最大クロック速度を超えないことを保証する。データは、シリアルデータラインに沿って、同時に1ビットずつ送受信される。チップは、パワーアップ時、パワーダウン時にリセットを実行する。その上、チップ内のタンパー検出及び防止回路は、攻撃が検出された場合、（検出された攻撃に応じて）チップをリセットさせるか、又はフラッシュメモリを消去させる。特殊プログラミングモードは、CLK電圧を特定レベルに保持することによってイネーブル状態にされる。これは次のセクションで更に説明され

る。

【1322】

論理的インタフェース

認証チップは、ノーマルモードとプログラミングモードの二つの動作モードを有する。二つのモードが必要になる理由は、動作プログラムコードは、（安全性の理由から）ROMではなくフラッシュメモリに記憶されるからである。プログラミングモードは、製造後にテスト目的のため使用され、動作プログラムコードを詰め込み、一方、ノーマルモードは、その後のチップの使用のために用いられる。

【1323】

プログラミングモード

プログラミングモードはCLKライン上に特定電圧を所定の時間だけ維持することによって有効にされる。チップがプログラミングモードに入るとき、全てのフラッシュメモリは消去される（全ての秘密鍵情報及びプログラムコードを含む）。認証チップは、次に、消去を検証する。消去が成功であるならば、認証チップは、新しいプログラムコードに対応した384バイトのデータを受け取る。バイトは、バイト₀からバイト₃₈₃の順に転送される。ビットは、ビット₀からビット₇の順に転送される。384バイト全部のプログラムコードがロードされると、認証チップはハングする。消去が成功しなかった場合、認証チップは全くデータをフラッシュメモリにロードすることなくハングする。チップがプログラムされた後、チップを再起動することができる。チップがCLKライン上の通常電圧でリセットされたとき、ノーマルモードに入る。

【1324】

ノーマルモード

認証チップは、プログラミングモードではないときには、ノーマルモードである。認証チップがノーマルモードでスタートアップしたとき（例えば、パワーアップリセット）、認証チップは、そのときフラッシュメモリのプログラムコード領域に格納されているプログラムを実行する。プログラムコードは、システムと認証チップの間に通信メカニズムを組み込み、システムからのコマンド及びデータを受け取り、出力値を生成する。認証チップはシリアル通信をするので、ビットは同時に1ビットずつ転送される。システムは、簡単なオペレーションコマンドセットによって認証チップと通信する。各コマンドは3ビットオペコードによって定義される。オペコードの解釈は、IsTrustedビット及びIsWrittenビットの現在値に依存する。

【1325】

以下のオペレーションが定義される。

【1326】

【表130】

Op	T	W	Mn	入 力	出 力	説 明
000	—	—	CLR	—	—	クリア
001	0	0	SSI	[160, 160, 160]	—	秘密情報選択
010	0	1	RD	[160, 160]	[256, 160]	Mを安全に読み出し
010	1	1	RND	—	[160, 160]	ランダム
011	0	1	WR	[256]	—	Mを書き込み
011	1	1	TST	[256, 160]	[1]	テスト
100	0	1	SAM	[32]	[32]	AccessMode セット
101	—	1	GIT	—	[1]	IsTrusted 獲得
110	—	1	SMT	[32]	—	MinTicks セット

Op = オペコード、T = IsTrusted 値、W = IsWritten 値、Mn = ニューモニック、[n] = パラメータに必要なビット数。

【1327】

この表に定義されていないコマンドは、NOP（ノーオペレーション）として解釈される。その例には、（IsTrusted値又はIsWritten値とは無関係である）オペコード110及び111と、IsWritten=0のときのSSI以外のオペコードと、が含まれる。RD及びRNDのオペコードは同じであり、WR及びTSTのオペコードも同じであることに注意する必要がある。オペコードの受け取り後に動かされる実際のコマンドは、（IsWrittenが1である限り）IsTrustedビットの現在値に依存する。IsTrustedビットがクリアされているところでは、RD及びWR関数が呼び出される。IsTrustedビットがセットされると、RND及びTST関数が呼び出される。2組のコマンドは、信頼されている認証チップと信頼されていない認証チップとの間で相互に排他的である。認証チップ上でコマンドを実行するため、（システムのような）クライアントは、コマンドオペコードと、その後続く、そのオペコードに必要な入力パラメータと、を送る。オペコードは、最下位ビットから最上位ビットまで送られる。例えば、SSIコマンドを送るため、ビット1、0及び0がこの順に送られる。各入力パラメータも同じ方法で送られ、最初に最下位ビットから送られ、最後に最上位ビットまでが送られる。戻り値はこの方法で読み出され、最初に最下位ビットが読まれ、最後に最上位ビットが読まれる。クライアントは獲得するビット数がわかっていなければならない。

10

【1328】

一部のケースでは、あるチップのコマンドからの出力ビットは、別のチップのコマンドへの入力ビットとしてそのまま供給される。この一例は、RNDコマンドとRDコマンドである。信頼された認証チップ上のRNDへの呼び出しからの出力ビットはシステムによって保持されなくてもよい。その代わりに、システムは、その出力ビットをそのまま信頼されない認証チップのRDコマンドの入力へ転送することができる。各コマンドの説明は、それがその場合であることを示している。各コマンドは後続のセクションで詳細に検討される。一部のアルゴリズムは、常駐レジスタがフラッシュメモリに保持されているので、特別に設計されている。

20

【1329】

レジスタ

認証チップ内のメモリは認証プロトコルによって要求される変数を記憶するため不揮発性メモリを含む。以下の不揮発性メモリ（フラッシュ）変数が定義される。

30

【1330】

【表 1 3 1】

変数名	サイズ (ビット単位)	説 明
M [0..15]	256	シリアル番号、メディア残量等のような状態データを含む 16 ワード(各々16ビット)
K ₁	160	認証中に R を変換するため使用される鍵
K ₂	160	認証中に M を変換するため使用される鍵
R	160	現在の乱数
Access Mode [0..15]	32	M [n] に対する 16 組の 2 ビット AccessMode 値
MinTicks	32	鍵ベース関数への呼び出しの間のクロックチックの最小数
SIWritten	1	セットされている場合、秘密鍵情報 (K ₁ , K ₂ 及び R) はチップに書き込まれている。クリアされている場合、秘密情報は未だ書き込まれていない。
IsTrusted	1	セットされている場合、RND 関数及び TST 関数を呼び出せるが、RD 関数及び WR 関数を呼び出せない。クリアされている場合、RND 関数及び TST 関数を呼び出せないが、RD 関数及び WR 関数を呼び出せる。
合計ビット数	802	

10

アーキテクチャ概要

このセクション・章は、認証チップの必要な機能を実装することができる専用 CPU の上位定義を与える。この CPU は汎用 CPU ではないことに注意する必要がある。認証ロジックを実装するためオーダーメードされている。WRITE、TST、RND 等のような認証チップのユーザが目にする認証コマンドは、CPU 命令セットでかかれた小型プログラムとして全て実装される。CPU は、32 ビットのアキュムレータ（殆どのオペレーションで使用される）と、多数のレジスタと、を含む。CPU は、認証ロジックを実装するため専用に使立てられた 8 ビット命令を含む。各 8 ビット命令は、典型的に、4 ビットのアペコードと、4 ビットのアペラントと、により構成される。

【1 3 3 1】

動作速度

内部クロック周波数リミッタユニットは、チップが所定の周波数よりも速い速度で動作することを防止する。この周波数は、製造中にチップに組み込まれ、変更不可である。この周波数は約 4 から 10 MHz であることが推奨される。

【1 3 3 2】

コンポジション及びブロック図

認証チップは以下のコンポーネントを含む。

【1 3 3 3】

20

30

【表 1 3 2】

ユニット名	CMOS タイプ	説 明
クロック周波数リミッタ	ノーマル	認証チップの動作周波数が特定最大周波数を超えないことを保証する。
オーバーアンダーパワー検出ユニット	ノーマル	電源が有効動作レンジを保つことを保証する。
プログラミングモード検出ユニット	ノーマル	ユーザによるプログラミングモードの入力を許可
ノイズ生成器	ノーマル	タンパー保護及び検出回路で使用する I_{dd} ノイズを生成
状態機械	ノーマル	チップの二つの動作モード(プログラミングモード及びノーマルモード) を制御する。これは、CPU の 2 個の動作サイクルを生成し、長いコマンド演算中に時間を稼ぎ、動作サイクル中にオペコード及びオペランドを記憶する。
I/O ユニット	ノーマル	外界とシリアル通信を行う。
ALU	ノンフラッシング	32 ビットアキュムレータ及び汎用数学演算子及び論理演算子を含む。
最小チックユニット	ノーマル (99%) ノンフラッシング (1%)	ある種の鍵ベースの演算の間に(カウントダウンによって) プログラマブル最小限遅延を引き起こす。
アドレス生成器ユニット	ノーマル (99%) ノンフラッシング (1%)	特定のオペランドによる要求に応じて直接アドレス、間接アドレス、及び、インデックス付きアドレスを生成する。
プログラムカウンタユニット	ノーマル	9 ビット PC(プログラムカウンタ)、並びに、分岐及びサブルーチン制御用のロジックを含む。
メモリユニット	ノンフラッシング	9 ビットのアドレスによってアドレス指定される。8 ビット幅のプログラムフラッシュメモリと、32 ビット幅のフラッシュメモリと、RAM、ルックアップテーブルと、を含む。プログラミングコードのローディングを可能にさせるためプログラミングモード回路を更に含む。

10

20

30

図 1 8 1 は認証チップの概略ブロック図である。タンパー防止及び検出回路は図示されない。雑音発生器、過小過大電力検出ユニット、及びプログラミングモード検出ユニットは、タンパー防止及び検出回路に接続され、残りのユニットへは接続されない。

【1 3 3 4】

メモリマップ

図 1 8 2 はメモリマップの一例を示す図である。認証チップは外部メモリを持たないが、内部メモリを有する。内部メモリは 9 ビットによってアドレス指定され、32 ビット幅又は 8 ビット幅である(アドレスに依存する)。32 ビット幅メモリは不揮発性データ、HAMC-SHA1 のため使用される変数、及び定数を保持するため使用される。8 ビット幅メモリは、プログラムと、プログラムによって使用される様々なジャンプテーブルと、を保持するため使用される。アドレス分割(予備メモリレンジを含む)はアドレス生成及び復号化を最適化するため設計される。

40

【1 3 3 5】

定数

図 1 8 3 は定数メモリマップの一例の説明図である。定数領域は 32 ビット定数により構成される。これらは、(32 ビットの全部が 0、及び 32 ビットの全部が 1 のような)単純な定数、HAMC アルゴリズムによって使用される定数、並びに、SHA-1 アルゴリズムで使用するため必要な定数 $y_0 - 3$ 及び $h_0 - 4$ である。これらの値はリセットによって影響を受けない。定数を利用する唯一のオペコードは LDK である。このケースでは、オペランド及びメモリ配置は、アドレス生成及び復号化を最小限に抑えるため、密接に

50

関連付けられている。

【1336】

R A M

図184はR A Mメモリマップの一例の説明図である。R A M領域は、認証チップの汎用機能のために必要な32個のパリティチェック付き32ビットレジスタを含むが、チップの動作中だけである。R A Mは揮発性メモリであり、一旦電源が取り外されると、値は失われる。實際上、メモリは、(メモリ残留のために)その値をパワーダウン後のある時間は維持するが、パワーアップ後に利用可能であるとは考えられない。これは、この文書の他のセクションで記載されているセキュリティの問題である。R A Mは、H M A C - S H A 1 アルゴリズムで使用される変数、即ち、A - E、一時変数T、160ビット作業用ハッシュ値のためのスペースH、(H M A Cによって要求される)ハッシュ結果の一時記憶のためのスペースB 160、及び拡張ハッシングメモリの512ビット用のスペースXを含む。全てのR A M変数は、リセット後に0にクリアされるが、プログラムコードはこれを当てにしてはならない。R A Mアドレスを利用するオペコードは、L D、S T、A D D、L O G、X O R及びP R Lである。全てのケースで、オペランド及びメモリ配置は、アドレス生成及び復号化を最小限に抑えるため、密接に関連付けられている(マルチワード変数は最上位ワードが先に格納される)。

10

【1337】

フラッシュメモリー変数

図185はフラッシュメモリー変数メモリマップの一例の説明図である。フラッシュメモリー領域は、認証チップ内の不揮発性情報を格納する。フラッシュメモリーは、電源の取り外し後もその値を維持し、次に電源が入れられたとき、変化していないことが期待できる。マルチステートフラッシュメモリーに保持された不揮発性情報は、2個の160ビット鍵(K_1 及び K_2)と、現在ランダム値(R)と、状態データ(M)と、M i n T i c k s 値(MT)と、A c c e s s M o d e 値(AM)と、I s W r i t t e n (I S W)フラグと、I s T r u s t e d (I S T)フラグと、を含む。フラッシュ値は、リセットによって変わらないが、プログラミングモードに入ったときに(0に)クリアされる。フラッシュアドレスを利用するオペレーションは、L D、S T、A D D、P R L、R O R、C L R及びS E Tである。全てのケースで、オペランド及びメモリ配置は、アドレス生成及び復号化を最小限に抑えるため、密接に関連付けられている。マルチワード変数 K_1 、 K_2 及びMは、アドレッシングの必要条件のため、最上位ワードから先に格納される。使用されるアドレッシングスキームは、Nで始まり、0で終わるインデックスによるベースアドレスオフセットである。このようにして、 M_N は最初にアクセスされるワードであり、 M_0 は、ループ処理でアクセスされる最後の32ビットワードである。マルチワード変数Rは、同じインデキシングスキームを使用するL F S R生成を簡単にするため、最下位ワードから先に格納される。

20

30

【1338】

フラッシュメモリープログラム

図186はフラッシュメモリーのプログラムメモリマップの一例の説明図である。第2のマルチステートフラッシュメモリー領域は、 384×8 ビットである。この領域は、J S R、J S I及びT B R命令のためのアドレステーブルと、D B Rコマンド用のオフセットと、定数と、プログラム自体と、を格納する。フラッシュメモリーは、リセットによって変わらないが、プログラミングモードに入ったときに(0に)クリアされる。一旦、プログラミングモードに入ると、8ビットフラッシュメモリーに、新しい384バイトの組をロードすることができる。これが終了すると、チップはリセットされ、通常のチップ動作が行われる。

40

【1339】

レジスタ

多数のレジスタが認証チップに定義される。それらは、関数実行中に一時記憶として使用される。一部は算術関数用であり、他の一部はカウンティング及びインデキシングに使用

50

され、その他はシリアル I / O のため使用される。これらのレジスタは不揮発性（フラッシュ）メモリに保持しなくてもよい。それらは、（フラッシュメモリとは異なり）消去サイクル無しで読み出し又は書き込みをすることができる。秘密情報を格納する一時記憶レジスタは、タンパー防止及び検出回路並びにパリティチェックによって、物理的攻撃から保護されなければならない。全てのレジスタはリセット後に 0 にクリアされる。しかし、プログラムコードは特定の状態を前提とすべきではなく、レジスタ値を適切に設定すべきである。これらのレジスタは、タンパー防止及び検出回路のために定義された様々な OK ビットを含まなくてもよいことに注意する必要がある。OK ビットは種々のユニットに散在され、リセット後に 1 にセットされる。

【1340】

サイクル

1 ビットサイクル値は、CPU がフェッチサイクル（0）であるか、実行サイクル（1）であるかを決定する。サイクルは、実際には、前のサイクル値を保持する 1 ビットレジスタから得られる。サイクルは命令セットから直接アクセスできない。それは内部レジスタ専用である。

【1341】

プログラムカウンタ

6 レベルの深さの 9 ビットプログラムカウンタアレイ（PCA）が定義される。それは、3 ビットスタックポインタ（SP）によってインデキシングされる。現在プログラムカウンタ（PC）は、現在実行中命令のアドレスを格納し、事実上、PCA [SP] である。更に、9 ビットAdrレジスタが定義され、（インデックス付きアクセス、又は間接メモリアクセスのための）現在メモリ基準の分解アドレスを格納する。PCA、SP 及びAdrレジスタは、命令セットから直接アクセスできない。これらは内部レジスタ専用である。

【1342】

CMD

8 ビットCMDレジスタは、現在実行中のコマンドを保持するため使用される。CMDレジスタレジスタは命令セットから直接アクセス不可であるが、内部レジスタ専用である。

【1343】

アキュムレータ及び Z フラグ

アキュムレータは 32 ビット汎用レジスタである。これは、算術演算への入力の一つとして使用され、メモリレジスタ間で情報を転送するため使用されるレジスタである。Z レジスタは 1 ビットフラグであり、アキュムレータが書き込まれるときに更新される。Z レジスタはアキュムレータの零性を格納する。アキュムレータに最後に書き込まれた値が 0 であるならば、Z = 1 であり、最後に書き込まれた値が非 0 であるならば、Z = 0 である。アキュムレータと Z レジスタの両方は命令セットから直接アクセス可能である。

【1344】

カウンタ

多数の専用カウンタ／インデックスレジスタが定義される。

【1345】

【表 133】

名 前	レジスタ サイズ	ビット数	説 明
C1	1×3	3	配列 : AE, B160, M, H, y, 及び h をインデックス付けするため使用されるカウンタ
C2	1×5	5	汎用カウンタ
N _{1..4}	4×4	16	配列 X をインデックス付けするため使用

これらの全てのカウンタ・レジスタは、命令セットから直接アクセス可能である。これらに特定の値をロードするための専用命令が存在し、別の命令は、それらをデクリメント又

はインクリメントし、或いは、特定のカウンタが零であるかどうかに応じて分岐する。2 個の専用フラグ（レジスタではない）が C 1 及び C 2 に関連付けられ、これらのフラグは C 1 又は C 2 の零性を保持する。これらのフラグは、ループ制御のために使用され、以下に列挙される。これらのフラグはレジスタではないが、レジスタと同様にテストできる。

【 1 3 4 6 】

【表 1 3 4】

名 前	説 明
C1Z	1 = C1 は現在零、0 = C1 は現在非零。
C2Z	1 = C2 は現在零、0 = C2 は現在非零。

10

フラグ

C P U 動作モードに対応する多数の 1 ビットフラグが定義される。

【 1 3 4 7 】

【表 1 3 5】

名 前	ビット数	説 明
WE	1	X レジスタ配列用書き込みイネーブル 0 = X レジスタへの書き込みはノップになる。 1 = X レジスタへの書き込みが実行される。
K2MX	1	0 = K1 は K 参照中にアクセスされる。M からの読み出しは 0 の読み出しとして解釈される。 1 = K2 は K 参照中にアクセスされる。M からの読み出しは成功。

20

これらの全ての 1 ビットフラグは命令セットから直接アクセス可能である。専用命令がこれらのフラグをセット及びクリアするため設けられる。

【 1 3 4 8 】

書き込み完全性のため使用されるレジスタ

【 1 3 4 9 】

【表 1 3 6】

名 前	ビット数	説 明
EE	1	WR コマンド擬似コードの EqEncountered 変数に対応する。上位の全てのコンポーネントが前の値と一致するかどうかを判定するため多倍精度データ値の書き込み中に使用される。
DE	1	WR コマンド擬似コードの DecEncountered 変数に対応する。上位の全てのコンポーネントが既にデクリメントされたかどうかを判定するため多倍精度データ値の書き込み中に使用される。

40

I / O のため使用されるレジスタ

4 個の 1 ビットレジスタがクライアント（システム）と認証チップの間の通信のため定義される。これらのレジスタは、InBit（入力ビット）、InBitValid（入力ビット有効）、OutBit（出力ビット）、及びOutBitValid（出力ビット有効）である。InBit及びInBitValidは、クライアントがコマンド及びデータを認証チップへ送るための手段を提供する。OutBit及びOutBitValidはクライアントが認証チップから情報を受けるための手段を提供する。クライアントは、コマンド及びパラメータビットを、同時に 1 ビットずつ認証チップへ送る。認証チップはスレーブ装置であるため、認証チップの観点では：

InBitからの読み出しは、InBitValidがクリアされている間はハングする。InBitValidは

50

、クライアントが次の入力ビットをInBitに書き込むまでクリアされたままである。InBitの読み出しはInBitValidビットをクリアし、次のInBitをクライアントから読み出せるようになる。クライアントは、InBitValidビットがクリアされるまでビットを認証チップに書き込めない；

OutBitへの書き込みは、OutBitValidがセットされている間はハングする。OutBitValidは、クライアントがOutBitからビットを読み出すまでセットされたままである。OutBitの書き込みはOutBitValidビットをセットし、次のOutBitをクライアントが読み出せるようにする。クライアントは、OutBitValidビットがセットされるまで認証チップからビットを読み出せない。

【1350】

10

タイミングアクセスのため使用されるレジスタ

1個の32ビットレジスタがタイマーとして使用するため定義される。MTR (MinT i c k s R e m a i n i n g) レジスタは、命令が実行されると、その都度デクリメントする。MTRレジスタが0になると、それは零に留まる。MTRは1ビットフラグMTR Zに関連付けられ、MTR ZはMTRの零性を格納する。MTR Zが1であるならば、MTRレジスタは零である。MTR Zが0であるならば、MTRレジスタは未だ零ではない。MTRは（リセット後、又は特定の鍵アクセス関数後）常にMinT i c k s 値から始まり、最終的に0まで減少する。MTRをセットし、MTR Zを専用命令でテストすることができるが、MTRの値は命令によって直接読み出せない。

【1351】

20

レジスタの要約

以下のテーブルは（レジスタ名の順に）全ての一時レジスタを要約する。そのテーブルは、レジスタ名と、サイズ（ビット単位）と、その指定されたレジスタがどこにあるかを列挙する。

【1352】

【表 1 3 7】

レジスタ名	ビット数	パリティ	検出場所
Acc	32	1	算術論理ユニット
Adr	9	1	アドレス生成器ユニット
AMT	32		算術論理ユニット
C1	3	1	アドレス生成器ユニット
C2	5	1	アドレス生成器ユニット
CMD	8	1	状態機械
Cycle (Old = prev Cycle)	1		状態機械
DE	1		算術論理ユニット
EE	1		算術論理ユニット
InBit	1		入出力ユニット
InBitValid	1		入出力ユニット
K2MX	1		アドレス生成器ユニット
MTR	32	1	最小チェックユニット
MTRZ	1		最小チェックユニット
N [1-4]	16	4	アドレス生成器ユニット
OutBit	1		入出力ユニット
OutBitValid	1		入出力ユニット
PCA	54	6	プログラムカウンタユニット
RTMP	1		算術論理ユニット
SP	3	1	プログラムカウンタユニット
WE	1		メモリユニット
Z	1		算術論理ユニット
合計ビット数	206	17	

10

20

30

命令セット

C P U は、認証ロジックを実現するため特別に用意された 8 ビット命令で動作する。8 ビット命令の大半は、4 ビットオペコードと、4 ビットオペランドと、により構成される。上位 4 ビットはオペコードを含み、下位 4 ビットがオペランドを含む。

【1 3 5 3】

オペコード及びオペランド（要約）

オペコードは以下の表に要約されている。

【1 3 5 4】

【表 1 3 8】

オペコード	ニューモニック	簡単な説明
0000	TBR	テスト及び分岐
0001	DBR	デクリメント及び分岐
001	JSR	テーブルによるサブルーチンジャンプ
01000	RTS	サブルーチンからのリターン
01001	JSI	間接サブルーチンジャンプ
0101	SC	カウンタセット
0110	CLR	特定のフラッシュレジスタクリア
0111	SET	特定のフラッシュレジスタのビットセット
1000	ADD	32 ビット値をアキュムレータに加算
1001	LOG	論理演算 (AND 及び OR)
1010	XOR	ある値によるアキュムレータの排他的論理和
1011	LD	特定の場所からのアキュムレータへのロード
1100	ROR	アキュムレータの右回転
1101	RPL	ビット置換
1110	LDK	アキュムレータに定数をロード
1111	ST	アキュムレータを指定された場所に保存

10

20

以下の表は、どのオペランドがどのオペコードと共に使用できるかを要約した表である。この表は、オペコードのニューモニックによってアルファベット順に並べられている。各オペランドのバイナリ値は、その次の表に示されている。

【 1 3 5 5】

【表 1 3 9】

オペコード	有効オペランド
ADD	{A, B, C, D, E, T, MT, AM, AE [C1], B160 [C1], H [C1], M [C1], K [C1], R [C1], X [N4]}
CLR	{WE, K2MX, M [C1], Group1, Group2}
DBR	{C1, C2}, DBR Table へのオフセット
JSI	{}
JSR	Table1 へのオフセット
LD	{A, B, C, D, E, T, MT, AM, AE [C1], B160 [C1], H [C1], M [C1], K [C1], R [C1], X [N4]}
LDK	{0x0000..., 0x3636..., 0x5C5C..., 0xFFFF, h [C1], y [C1]}
LOG	{AND, OR}, {A, B, C, D, E, T, MT, AM}
ROR	{InBit, OutBit, LFSR, IST, ISW, MTRZ, 1, 2, 27, 31}
RPL	{Init, MHI, MLO}
RTS	{}
SC	{C1, C2}, カウンタリストへのオフセット
SET	{WE, K2MX, Nx, MTR, IST, ISW}
ST	{A, B, C, D, E, T, MT, AM, AE [C1], B160 [C1], H [C1], M [C1], K [C1], R [C1], X [N4]}
TBR	{0, 1}, Table1 へのオフセット
XOR	{A, B, C, D, E, T, MT, AM, X [N1], X [N2], X [N3], X [N4]}

30

40

以下のオペランド表は、4 ビットオペランドの解釈を示し、全ての4ビットが直接解釈のため使用されている。

【 1 3 5 6】

50

【表 1 4 0】

オペランド	ADD, LD, ST	XOR	ROR	LDK	RPL	SET	CLR
0000	E	E	InBit	0x00...	Init	WE	WE
0001	D	D	OutBit	0x36...	—	K2MX	K2MX
0010	C	C	RB	0x5C...	—	Nx	—
0011	B	B	XRB	0xFF...	—	—	—
0100	A	A	IST	y [C1]	—	IST	—
0101	T	T	ISW	—	—	ISW	—
0110	MT	MT	MTRZ	—	—	MTR	—
0111	AM	AM	I	—	—	—	—
1000	AE [C1]	—	—	h [C1]	—	—	—
1001	B160 [C1]	—	2	—	—	—	—
1010	H [C1]	—	27	—	—	—	—
1011	—	—	—	—	—	—	—
1100	R [C1]	X [N1]	31	—	—	—	R
1101	K [C1]	X [N2]	—	—	—	—	Group1
1110	M [C1]	X [N3]	—	MLO	—	—	M [C1]
1111	X [N4]	X [N4]	—	MHI	—	—	Group2

以下の命令は、オペランドの最上位ビットに基づいて選択を行う。

【 1 3 5 7 】

【表 1 4 1】

オペランド ₃	どのカウンタか? (DBR, SC)	どの演算か? (LOG)	どの値か? (TBR)
0	C1	AND	Zero
1	C2	OR	Non-zero

オペランドの下位 3 ビットは、オフセット (DBR, TBR) か、専用テーブル (SC) からの値であるか、LOG の場合のように、それらは論理演算のための第 2 の入力を選択する。解釈は、ADD、LD、及び ST オペコードの解釈と一致する。

【 1 3 5 8 】

【表 1 4 2】

オペランド _{2:0}	LOG 入力 2	SC 値
000	E	2
001	D	3
010	C	4
011	B	7
100	A	10
101	T	15
110	MT	19
111	AM	31

ADD—アキュムレータへの加算

ニューモニック： ADD

オペコード： 1 0 0 0

用法： ADD

値。

【1359】

ADD命令は、指定されたオペランドを、モジュロ 2^{32} の加法でアキュムレータに加算する。オペランドは、A、B、C、D、E、T、AM、MT、AE[C1]、H[C1]、B160[C1]、R[C1]、K[C1]、M[C1]、又はX[N4]のうちの一つである。また、Zフラグはこの演算中に、ロードされた値が零か非零かに応じてセットされる。

【1360】

CLR—ビットをクリア
 ニューモニック： CLR
 オペコード： 0110
 用法： CLR
 フラグ／レジスタ。

10

【1361】

CLR命令は、指定された内部フラグ、又はフラッシュメモリレジスタをクリアさせる。フラッシュメモリのケースでは、CLR命令はある時間を要するが、次の命令は、フラッシュメモリの消去が終了するまで引き延ばされる。クリアできるレジスタは、WE及びK2MXである。クリアできるフラッシュメモリは、R、M[C1]、Group1、及びGroup2である。Group1は、IST及びISWフラグである。それらがクリアされると、唯一の有効な上位レベルコマンドはSSI命令である。Group2は、MT、AM、K1及びK2レジスタである。Rは別個に消去される。なぜならば、RはTSTへの呼び出し毎に更新されるからである。また、Mは、インデックスメカニズムによって消去され、Mの個々の部分を更新することができる。また、対応したSET命令がある。

20

【1362】

DBR—デクリメント及びブランチ
 ニューモニック： DBR
 オペコード： 0001
 用法： DBR
 カウンタ、オフセット。

【1363】

この命令は、簡単なループを構築するメカニズムを提供する。オペランドのハイビットは、C1のテストか、C2のテスト（2個のカウンタ）を選択する。指定されたカウンタが非零であるならば、カウンタはデクリメントされ、所定のオフセットの値（符号拡張）がPCに加算される。指定されたカウンタが零であるならば、カウンタはデクリメントされ、処理はPC+1で継続する。8エンタリーのオフセットテーブルがアドレス0110000000（プログラムメモリの64番目のエンタリー）に格納される。オフセットの8ビットは符号付き数として取り扱われる。このようにして、0xFFは-1として扱われ、0x01は+1として取り扱われる。典型的に、その値はループで使用するため負である。

30

【1364】

JSI—サブルーチン間接ジャンプ
 ニューモニック： JSI
 オペコード： 01001
 用法： JSI (Acc)。

40

【1365】

JSI命令は、アキュムレータの現在値に依存したサブルーチンへのジャンプを可能にさせる。この命令は、現在PCをスタックへプッシュし、PCに新しい値をロードする。新しいPCの上位8ビットはジャンプテーブル2からロードされ（オフセットはアキュムレータの下位5ビットによって与えられる）、PCの最下位ビットは0にクリアされる。このようにして、全てのサブルーチンは偶数アドレスでスタートしなければならない。スタックは、6個の実行レベル（5回のサブルーチン深さ）を提供する。プログラマは、こ

50

の深さを超えないこと、又は戻り値が上書きされること（スタックがラップするため）を保証する責任がある。

【1366】

JSR—サブルーチンジャンプ

ニューモニック： JSR

オペコード： 001

用法： JSR

オフセット。

【1367】

JSR命令は、最も一般的なサブルーチン構造の用法を提供する。この命令は、現在PCをスタックへプッシュし、PCに新しい値をロードする。新しいPCの上位8ビットはアドレステーブル1から与えられ、テーブルへのオフセットは5ビットオペランド（32通りのアドレスを実現できる）によって与えられる。PCの最下位ビットは0にクリアされる。このようにして、全てのサブルーチンは偶数アドレスでスタートしなければならない。スタックは、6個の実行レベル（5回のサブルーチン深さ）を提供する。プログラマは、この深さを超えないこと、又は戻り値が上書きされること（スタックがラップするため）を保証する責任がある。

10

【1368】

LD—アキュムレータへのロード

ニューモニック： LD

オペコード： 1011

用法： LD

値。

20

【1369】

LD命令は、指定されたオペランドからアキュムレータにロードする。オペランドは、A、B、C、D、E、T、AM、MT、AE[C1]、H[C1]、B160[C1]、R[C1]、K[C1]、M[C1]、又はX[N4]のうちの一つである。また、Zフラグはこの演算中に、ロードされた値が零か非零かに応じてセットされる。

【1370】

LDK—定数のロード

ニューモニック： LDK

オペコード： 1110

用法： LDK

定数。

30

【1371】

LDK命令は、アキュムレータに指定された定数をロードする。定数は、HMACHA1のために必要な32ビット値と、汎用処理のために最も有用な全て0及び全て1と、である。したがって、

0x00000000

0x36363636

0x5C5C5C5C

0xFFFFFFF

40

又は、C1でインデックス付けされたh及びy定数テーブルからの選択肢がある。hとyの定数テーブルは、HMACHA1に必要な32ビットの表形式の定数を保持する。また、Zフラグはこの演算中に、ロードされた値が零か非零かに応じてセットされる。

【1372】

LOG—論理演算

ニューモニック： LOG

オペコード： 1001

用法： LOG

50

演算 値。

【1373】

LOG命令は、アキュムレータと指定された値に対して、32ビットのビット論理演算を実行する。LOG命令によってサポートされる2個の演算は、AND及びORである。ビットNOT及びXOR演算は、XOR命令によってサポートされる。アキュムレータとAND演算又はOR演算される32ビット値は、A、B、C、D、E、T、MT及びAMのうちの一つである。また、Zフラグはこの演算中に、（アキュムレータにロードされる）得られた32ビット値が零か非零かに応じてセットされる。

【1374】

ROR—右回転

ニューモニック： ROR

オペコード： 1100

用法： ROR

値。

【1375】

ROR命令は、アキュムレータを設定されたビット数だけ右へ回転する方法を提供する。アキュムレータの最上部に現れるビット（ビット31になる）は、アキュムレータの前のビット0の値でもよく、又は（フラグ、又はシリアル入力コネクションのような）外部1ビットフラグからでもよい。回転して溢れたビットはシリアルコネクションから出力してもよく、又は外部フラグと結合してもよい。許容されるオペランドは、InBit、OutBit、LFSR、RLFSR、IST、ISW、MTRZ、1、2、27、及び31である。また、Zフラグはこの演算中に、（アキュムレータにロードされる）得られた32ビット値が零か非零かに応じてセットされる。最も簡単な形式では、ROR命令用のオペランドは、アキュムレータを回転させるビット位置数を示す1、2、27、31のうちの一つである。これらのオペランドの場合、外部入力又は出力はなく、アキュムレータのビットが単に右へ回転させられる。オペランドがIST、ISW及びMTRZの場合、適切なフラグがアキュムレータの最上位ビットへ転送される。アキュムレータの残りの部分は、1ビット位置だけ右へシフトされ（ビット31がビット30になる等）、アキュムレータの最下位ビットはシフトされて溢れる。オペランドがInBitである場合、次のシリアル入力ビットがアキュムレータの最上位ビットへ転送される。次に、InBitValidビットはクリアされる。クライアントから入力ビットが得られない場合、実行は入力ビットが現れるまで一時停止される。アキュムレータの残りの部分は、1ビット位置だけ右へシフトされ（ビット31がビット30になる等）、アキュムレータの最下位ビットはシフトされて溢れる。

【1376】

オペランドがOutBitである場合、アキュムレータは右へ1ビット位置だけシフトされる。ビット0からシフトで送り出されたビットはOutBitフラグに格納され、OutBitValidフラグがセットされる。したがって、それは、クライアントによる読み出しの準備が完了する。OutBitValidフラグが既にセットされている場合、命令の実行は、OutBitがクライアントによって読み出される（そして、OutBitValidフラグがクリアされる）まで引き延ばされる。ビット31へシフトされた新しいビットはガーベッジ（実際には、現在のInBitレジスタの値）であると考えられる。最後に、RB及びXBRオペランドは、LFSRと倍精度シフトレジスタの実装を可能にさせる。RBの場合、シフトで送り出されたビット（形式的にビット0）は、RTMPレジスタに書き込まれる。RTMPレジスタの現在のレジスタはアキュムレータの新しいビット31になる。数個の32ビット値に複数のRORRBコマンドを実行すると、倍精度の右回転／シフトが実現される。XRBは、RBと同様に動作し、RTMPレジスタの現在値がアキュムレータの新しいビット31になる。しかし、XRB命令の場合、形式的にビット0であることが分かっているビットは、（RB命令の用に）単純にRTMPを置き換えない。その代わりに、そのビットはXORとRTMP演算され、その結果がRTMPに格納される。これにより、認証プロトコルによって要求される長いLFSRを実現できる。

10

20

30

40

50

【1377】

RPLービット置換

ニューモニック： RPL

オペランド： 1101

用法： ROR

値。

【1378】

RPL命令は、上位レベルWRIT Eコマンドを認証チップに組み込むため設計された。この命令は、アキュムレータの上位16ビットを、（アクセスモード値に依存して）最終的にMアレイに書き込まれた値で置き換えることを予定している。この命令は、3個のオペランド：In it、MH I及びML Oをとる。In itオペランドは、全ての内部フラグをセットし、後続の処理のためALU内のRPLユニットを準備する。アキュムレータは内部Ac c e s s M o d eレジスタへ転送される。アキュムレータは、WRIT Eコマンドを実行する場合には、RPL

10

In itへの呼び出しの前に、AMフラッシュメモリ場所からロードされるべきであり、TSTコマンドを実行する場合には、0でロードされるべきである。アキュムレータはそのまま変化しない。MH I及びML Oオペランドは、M[C1]の上位16ビット又は下位16ビットが（常に）アキュムレータの上位16ビットとの比較に使用されるかどうかを参照する。実行されるMH I及びML O命令の各々は、初期化されたAc c e s s M o d e値からの引き続く2ビットを使用する。最初のMH I又はML Oの実行は最下位2ビットを使用し、次の実行はその次の2ビットを使用し、以下同様である。

20

【1379】

RTSーサブルーチンからのリターン

ニューモニック： RTS

オペコード： 01000

用法： RTS。

【1380】

RTS命令は、直前に実行されたJSR又はJSI命令の後の命令で実行を再開させる。したがって、サブルーチンからの復帰と呼ばれる。実際には、この命令は、セーブされたPCをスタックからプルし、1を加え、得られたアドレスから実行を再開する。6個の実行レベル（5回のサブルーチン）が提供されるが、JSR及びJSI命令の各々と、RTSを合わせることはプログラマの責任である。先行するJSR無しに実行されたRTSは、スタックからプルされたアドレスから実行を開始させる。

30

【1381】

SCーカウンタセット

ニューモニック： SC

オペコード： 0101

用法： SC

カウンタ 値。

【1382】

SC命令はカウンタに特定の値をロードするため使用される。オペランドは、カウンタC1とC2のうちのどちらにロードするかを決める。ロードされるべき値は、2、3、4、7、10、15、19及び31のうちの一つである。カウンタ値は、ループ処理及びインデックス処理のため使用される。C1とC2の両方は、（DBR命令と組み合わせられたとき）ループ構造のため使用されるが、一方、C1だけが倍精度変数の32ビット部分をインデックス付けするため使用できる。

40

【1383】

SETービットセット

ニューモニック： SET

オペコード： 0111

50

用法： S E T
フラグ／レジスタ。

【 1 3 8 4 】

S E T 命令は、特定のフラグ又はフラッシュメモリの設定を可能にさせる。また、対応した C L R 命令がある。W E オペランド及び K 2 M X オペランドの各々は、後の処理のため指定されたフラグをセットする。I S T オペランド及び I S W オペランドの各々は、フラッシュメモリの適切なビットをセットし、一方、M T R オペランドはアキュムレータの現在値を M T R レジスタへ転送する。S E T N x コマンドは、N 1 から N 4 に以下の定数をロードする。

【 1 3 8 5 】

【表 1 4 3】

インデックス	ロードされた定数	参照される初期 X [N]
N1	2	X [13]
N2	7	X [8]
N3	13	X [2]
N4	15	X [0]

参照される各初期 X [N_n] は、最適化された S H A - 1 アルゴリズムのインデックス N 1 から N 4 に対する初期状態と一致することに注意すべきである。各インデックス値 N n がデクリメントするとき、有効な X [N] がインクリメントする。なぜならば、X 個のワードは、最上位ビットの方からメモリに格納されるからである。

【 1 3 8 6 】

S T - アキュムレータを保存

ニューモニック： S T

オペコード： 1 1 1 1

用法： S T

ロケーション。

【 1 3 8 7 】

S T 命令は、アキュムレータの現在値を指定されたロケーションに格納する。ロケーションは、A、B、C、D、E、T、AM、MT、A E [C 1]、H [C 1]、B 1 6 0 [C 1]、R [C 1]、K [C 1]、M [C 1]、又は X [N 4] のうちの一つである。X [N 4] オペランドは N 4 インデックスを進める副作用がある。格納が行われた後、N 4 は X アレイの次の要素を指示する。N 4 は 1 ずつデクリメントされるが、X アレイはハイからローへ並べられているので、インデックスをデクリメントすると、アレイ内の次の要素へ進む。格納先がフラッシュメモリである場合、S T 命令の効果は、アキュムレータ内のビットに対応したフラッシュメモリのビットがセットされることである。アキュムレータからの正確な値が格納されることを保証するために、適切なメモリロケーションを先に消去する C L R 命令を確実に使用すべきである。

【 1 3 8 8 】

T B R - テスト及びブランチ

ニューモニック： T B R

オペコード： 0 0 0 0

用法： T B R

値 インデックス。

【 1 3 8 9 】

テスト及びブランチ命令は、アキュムレータが零か非零かをテストし、アキュムレータの現在状態がテスト相手の状態と一致しているならば、所与のアドレスへ分岐する。Z フラグが T R B テストと一致するならば、P C を 9 ビット値で置き換え、ここで、b i t 0 = 0 であり、上位 8 ビットは M U から与えられる。そうでなければ、P C を 1 だけインクリ

10

20

30

40

50

メントする。値オペランドは0又は1である。0は、アキュムレータが零であることについてのテストであることを示す。1は、アキュムレータが非零であることについてのテストであることを示す。インデックスオペランドは、テストが成功したときに実行をジャンプさせる場所を示す。オペランドインデックスの残りの3ビットはジャンプテーブル1の最下位8エントリーをインデックス付けする。上位8ビットはテーブルから取り出され、最下位ビット（ビット0）は0にクリアされる。CMDはリセット時に0にクリアされる。0はTBR0として翻訳され、これは、アキュムレータ=0である場合に、アドレスオフセット0に格納されたアドレスへ分岐することを意味する。アキュムレータ及びZフラグもリセット時にクリアされるので、テストが真であり、正味の効果はジャンプテーブルの0番目のエントリーに格納されたアドレスへのジャンプである。

10

【1390】

XOR－排他的論理和

ニューモニック： XOR

オペコード： 1010

用法： XOR

値。

【1391】

XOR演算は、アキュムレータとの32ビットのビットXORを実行し、結果をアキュムレータに格納する。オペランドは、A、B、C、D、E、T、AM、MT、X[N1]、X[N2]、X[N3]又はX[N4]のうちの一つである。また、Zフラグは、結果に応じて（即ち、アキュムレータにロードされた値に応じて）、この演算中にセットされる。ビットNOT演算は、（LDK命令を用いて）アキュムレータを0xFFFFFFFFとXOR演算することによって実行される。X[N]オペランドは、（演算後に）適切なインデックスを次の値へ進める副作用がある。XORが行われた後、インデックスはXアレイの次の要素を指示する。N4は、ST

20

X[N4]命令によって進められる。インデックスは1ずつデクリメントされるが、Xアレイはハイからローへ並べられているので、インデックスをデクリメントすると、アレイ内の次の要素へ進む。

【1392】

プログラミングモード検出ユニット

30

プログラミングモード検出ユニットは入力クロック電圧を監視する。クロック電圧が特定の値であるならば、消去タンパー検出ラインはトリガーされ、全ての鍵、プログラムコード、秘密情報等を消去し、プログラムモードに入る。プログラミングモード検出ユニットは通常のCMOSを用いて実装可能である。なぜならば、鍵はこのユニットを通過しないからである。このユニットはノンフラッシングCMOSで実現しなくてもよい。プログラミングモード検出ユニットをタンパー検出ラインで覆うべき特別の必要性はない。なぜならば、攻撃者は、CLR入力によってチップをプログラミングモードに移すことができるからである。プログラミングモードへ入るための信号として、消去タンパー検出ラインを使用することは、攻撃者が攻撃の一部としてプログラミングモードを使用したい場合に、消去タンパー検出ラインをアクティブにし、且つ機能させなければならないことを意味する。これは、認証チップへの攻撃を非常に困難にさせる。

40

【1393】

雑音発生器

雑音発生器は通常のCMOSで実現できる。なぜならば、鍵はこのユニットを通らないからである。このユニットはノンフラッシングCMOSで実現しなくてもよい。しかし、雑音発生器は、両方のタンパー検出及び保護ラインで保護されるべきであり、この結果として、攻撃者がこのユニットを改ざん使用とする場合、チップは、リセットするか、又は全ての秘密情報をクリアする。その上、LFSRのビットは、それらが改ざんされていないことを保証するため検証されるべきである（即ち、パリティチェック）。パリティチェックが失敗した場合、消去タンパー検出ラインがトリガーされる。最後に、雑音発生器の6

50

4ビット全部が単一ビットにOR演算される。このビットが0であるならば、消去タンパー検出ラインがトリガーされる。なぜならば、0はLFSRの無効状態だからである。OKビットセットアップを使用することは意味がない。なぜならば、雑音発生器はタンパー検出及び防止回路だけによって使用されるからである。

【1394】

状態機械

状態機械は、CPUの二つの動作サイクルを生成し、長いコマンドオペレーション中に引き延ばしを行い、オペレーティングサイクル中にオペコード及びオペランドを記憶する。状態機械は通常のCMOSで実現可能であり、その理由は、鍵がこのユニットを通らないからである。このユニットはノンフラッシングCMOSで実現しなくてもよい。しかし、オペコード／オペランドのラッチは、パリティチェックされるべきである。状態機械に格納されるロジック及びレジスタは、両方のタンパー検出ラインによって覆われるべきである。これは、実行される命令が攻撃者によって変更されないことを保証する。

10

【1395】

認証チップは、汎用CPUの高速性及びスループットを要求しない。それは、認証プロトコルを実行できる速さで動作すればよく、それ以上に高速である必要はない。分岐制御を最適化し、又は次のオペコードをフェッチしている間にオペコードを実行する専用回路（及び、それに関連した全ての複雑さ）を具備するよりは、状態機械は、単純化した考え方を採用する。これは、設計時間を最小限に抑え、実装にエラーが起こる可能性を低減することに役立つ。

20

【1396】

状態機械の一般的な動作は、以下のサイクルの組を生成することである：

サイクル0：フェッチサイクル。これは、オペコードがプログラムメモリからフェッチされ、フェッチされたオペコードから実効アドレスが生成されるサイクルである；

サイクル1：実行サイクルこれは、オペランドが（サイクル0で）生成された実効アドレスによってルックアップされ（かもしれない）、オペレーション自体が実行されるサイクルである。

【1397】

正常な条件下で、状態機械は、サイクル：0、1、0、1、0、1...を生成する。しかし、あるケースでは、状態機械は、引き延ばしを行い、引き延ばし条件が終了するまで、各クロックチックでサイクル0を生成する。引き延ばし条件には、フラッシュサイクルの消去サイクル待ち、クライアントのシリアルデータの読み書きの待機、又は（タンパーリングによる）無効オペコード等が含まれる。フラッシュメモリがそのとき消去中であるならば、次の命令は、フラッシュメモリが消去され尽くすまで実行できない。これは、メモリユニットから送られるWait信号によって判定される。Wait=1であるならば、状態機械はサイクル0だけを生成する。シリアルI/O動作を原因とする引き延ばしには、2通りのケースがある：

30

オペコードがROR

OutBitであり、既にOutBitValid=1であるケース。これは、現在オペレーションは、ビットをクライアントへ出力することを要求するが、クライアントは最後のビットを未だ読み出していないケースである；

40

オペレーションがROR

InBitであり、InBitValid=0であるケース。これは、現在オペレーションがクライアントからのビットの読み出しを要求するが、クライアントが未だそのビットを供給していないケースである。

【1398】

両方のケースで、状態機械は、引き延ばし条件が終了するまで引き延ばしをしなければならない。したがって、次の「サイクル」は、古いサイクル又は前のサイクルと、CMD、Wait、OutBitValid、及びInBitValidの現在値と、に依存する。WaitはMUに由来し、OutBitValid及びInBitValidはI/Oユニットに由来する。サイクルが0であると

50

き、8ビットオペコードはメモリユニットからフェッチされ、8ビットCMDレジスタに収容される。CMDレジスタの書き込みイネーブルは、一サイクルである。このユニットからの二つの出力は、サイクルとCMDである。両方の値は、認証チップ内の他の全ての処理ユニットへ渡される。1ビットのサイクル値は、各ユニットにフェッチ又は実行サイクルが行われているかどうかを報せ、一方、8ビットのCMDは、各ユニットが特定のユニットに関連したコマンドのための適切な動作を行えるようにする。

【1399】

図187は、状態機械のコンポーネント間のデータフロー及び関係を示す図であり、同図では、

【1400】

【表144】

10

Logic ₁ :	Wait OR ~ (Old OR ((CMD=ROR) 6 ((CMD=InBit AND -InBitValid) OR (CMD=OutBit AND OutBitValid))))

である。

【1401】

Old及びCMDは、共にリセットで0にクリアされる。この結果によって第1のサイクルは1であり、これは、0CMDを実行させる。0はTBR

0として翻訳され、即ち、Accumulator=0ならば、アドレスオフセット0に格納されたアドレスへ分岐する。アキュムレータもリセットで0にクリアされるので、テストは真であり、正味の効果はジャンプテーブルの0番目のエントリーに格納されたアドレスへのジャンプである。2個のVALユニットがそれらを通過する値を検証するため設計される。各VALは、両方のタンパー防止及び検出ラインに接続されたOKビットを含む。OKビットはリセットで1にセットされ、サイクル毎に両方のタンパー防止及び検出ラインからのChipOK値とOR演算される。OKビットはそのユニットを通過する各データビットとAND演算される。VAL₁の場合、チップが改ざんされたならば、実効サイクルは常に0である。このようにして、決してサイクル1にならないので、プログラムコードは実行されない。Oldが改ざんされたかどうかを検査する必要はない。なぜならば、攻撃者がOld状態を止めるならば、チップはそれ以上命令を実行しないからである。VAL₂の場合、実効8ビットCMD値は、チップ改ざんされたならば、常に0であり、これは、TBR

20

30

0命令である。これは、あらゆるプログラムコードの実効を停止する。VAL₂は、CMDが改ざんされていないことを保証するため、CMDからのビットにパリティチェックを行う。パリティチェックが失敗したならば、消去タンパー検出ラインがトリガーされる。

【1402】

I/Oユニット

I/Oユニットは外界とシリアル通信する。認証チップは、スレーブシリアル装置として動作し、クライアントからのシリアルデータを受け取り、コマンドを処理し、得られたデータをクライアントへシリアル送信する。I/Oユニットは通常のCMOSで実現できる。なぜならば、鍵がこのユニットを通らないからである。このユニットはノンフラッシングCMOSで実現しなくてもよい。更に、ラッチはパリティチェックしなくてもよい。なぜならば、攻撃者にとってそれらを破壊又は変更する利益がないからである。I/Oユニットは、タンパー検出ラインの何れかが破断された場合、0を出力し0を入力する。これは、攻撃者がリセット及び／又は消去回路を動作禁止状態にした場合に限り有効になる。なぜならば、タンパー検出ラインの何れかを破断することは、リセット又は全フラッシュメモリの消去を生じるからである。

40

【1403】

InBit、InBitValid、OutBit、及びOutBitValidの1ビットレジスタは、クライアント（システム）と認証チップとの間の通信のため使用される。InBit及びInBitValidは、クライ

50

アントがコマンド及びデータを認証チップへ送る手段を提供する。OutBit及びOutBitValidは、クライアントが認証チップから情報を得る手段を提供する。チップがリセットされたとき、InBitValid及びOutBitValidは共にクリアされる。クライアントは、同時に1ビットずつ、コマンド及びパラメータビットを認証チップへ送る。認証チップの観点では：InBitからの読み出しは、InBitValidがクリアされている間はハングする。InBitValidは、クライアントが次の入力ビットをInBitに書き込むまでクリアされたままである。InBitの読み出しはInBitValidビットをクリアし、次のInBitをクライアントから読み出せるようになる。クライアントは、InBitValidビットがクリアされるまでビットを認証チップに書き込めない；

OutBitへの書き込みは、OutBitValidがセットされている間はハングする。OutBitValidは、クライアントがOutBitからビットを読み出すまでセットされたままである。OutBitの書き込みはOutBitValidビットをセットし、次のOutBitをクライアントが読み出せるようにする。クライアントは、OutBitValidビットがセットされるまで認証チップからビットを読み出せない。

【1404】

実際のコマンドの引き延ばしは、状態機械によって管理されるが、種々の通信レジスタ及び通信回路がI/Oユニットに存在する。

【1405】

図188はI/Oユニットのコンポーネント間のデータフロー及び関係を示す図である。

【1406】

【表145】

Logic:	Cycle AND (CMD=ROR OutBit)

シリアルI/Oユニットは、データピンを使って外界と外部的に通信する回路を含む。InBitUsed制御信号は、所定のクロックサイクル中にInBitを使用する何れかのユニットによってセットされるべきである（サイクルのあらゆる状態になり得る）。2個のVALユニットは、タンパー防止及び検出回路に接続された検証ユニットであり、各VALはOKビットを含む。OKビットはリセットで1にセットされ、サイクル毎に両方のタンパー防止及び検出ラインからのChipOK値とOR演算される。OKビットはそのユニットを通過する各データビットとAND演算される。

【1407】

VAL₁の場合、チップが改ざんされたならば、チップからの実効ビット出力は常に0である。このようにして、攻撃者は有用な出力を生成できない。VAL₂の場合、チップ改ざんされたならば、チップへの実効ビット入力に常に0である。このようにして、攻撃者は有用な入力を選択できない。I/Oユニットのレジスタを検証する必要はない。なぜならば、攻撃者はそれらを破壊又は変更することによる利益がないからである。

【1408】

ALU

図189は、算術論理ユニットの概略ブロック図である。算術論理ユニット（ALU）は、32ビットAcc（アキュムレータ）レジスタと、簡単な算術及び論理演算用の回路を含む。ALU及び全てのサブユニットは、ノンフラッシングCMOSで実現されるべきである。なぜならば、鍵がその中を通るからである。その上、アキュムレータはパリティチェックされる。ALUに収容されたロジック及びレジスタは、両方のタンパー検出ラインによってカバーされる。これは、鍵及び中間計算値が攻撃者によって変更されないことを保証する。1ビットのZレジスタは、アキュムレータの零性を格納する。Z及びアキュムレータレジスタの両方は、リセット後にクリアされる。Zレジスタはアキュムレータが更新されたときに更新され、アキュムレータは、LD、LDK、LOG、XOR、ROR、RPL及びADDの何れかのコマンドに対して更新される。各算術及び論理ブロックは、二つの32ビット入力、即ち、アキュムレータの現在値と、MUの現在32ビット出力と

、に基づいて動作する。ここで、

【1409】

【表146】

Logic ₁ :	Cycle AND CMD ₇ AND (CMD _{6:4} ≠ ST)

である。

【1410】

Acc及びZのWrite Enablesは、(Logic1により) CMD₇及びサイクルを考慮するので、これらの2ビットは、出力を選択するためにマルチプレクサMX₁により要求されない。MX₁の出力選択は、CMDのビット6-3だけを要求するので、結果としてより簡単である。

【1411】

【表147】

	出力	CMD ₆₋₃
MX ₁	ADD	ADD
	AND	LOG AND
	OR	LOG OR
	XOR	XOR
	RPL	RPL
	ROR	ROR
	MU から	LD 又は LDK

2個のVALユニットは、タンパー防止及び検出ラインに接続された検証ユニットであり、各VALはOKビットを含む。OKビットはリセットで1にセットされ、サイクル毎に両方のタンパー防止及び検出ラインからのChip OK値とOR演算される。OKビットはそのユニットを通過する各データビットとAND演算される。VAL₁の場合、チップが改ざんされたならば、アキュムレータから出力された実効ビットは常に0である。これは、攻撃者がアキュムレータに関する何かを処理することを妨げる。VAL₁はアキュムレータに対するパリティチェックを実行し、チェックが失敗したならば、消去タンパー検出ラインをセットする。VAL₂の場合、アキュムレータの実効Z状態は、チップ改ざんされたならば、常に真である。このようにして、攻撃者はループ構造を作成できない。ALUの残りの機能ブロックは後述する。すべてがフラッシュしないCMOSで実現されるべきである。

【1412】

【表 1 4 8】

ブロック	説 明
OR	マルチプレクサ MX_1 から 32 ビット出力を取得し、1 ビットを得るために 32 ビットの全てを一緒に論理和演算する。
ADD	2 入力の加算の結果をモジュロ 2^{32} で出力する。
AND	二つの 32 ビット入力の並列ビット論理積の 32 ビットの結果を出力する。
OR	二つの 32 ビット入力の並列ビット論理和の 32 ビットの結果を出力する。
XOR	二つの 32 ビット入力の並列ビット排他論理和の 32 ビットの結果を出力する。
RPL	詳細は後述する。
ROR	詳細は後述する。

10

R P L

図 1 9 0 は R P L ユニットの概略ブロック図である。R P L ユニットは A L U 内のコンポーネントである。これは、認証チップの R P L

C M P 機能を実現するため設計されている。R P L C M P コマンドは、A c c e s s M o d e の値に基づいて、フラッシュメモリ M への安全な書き込みに使用するため特に設計された。R P L ユニットは、シフトパルス毎に 2 ビット右シフトを実行する A M T (A c c e s s M o d e T e m p) と呼ばれる 3 2 ビットシフトレジスタと、W R 擬似コードの E q E n c o u n t e r e d フラグ及び D e c E n c o u n t e r e d フラグに直接基づいている E E 及び D E と呼ばれる 1 ビットレジスタと、を含む。全てのレジスタはリセット後に 0 にクリアされる。A M T は、P R L

20

I N I T コマンドを用いて (アキュムレータを介して) 3 2 ビット A M 値がロードされ、E E 及び D E は、R P L

M H I への呼び出し及び R P L

M L O への呼び出しを用いて汎用書き込みアルゴリズムに従ってセットされる。E Q ブロック及び L T ブロックは W R コマンド擬似コードに記載されている通りの機能を備えている。E Q ブロックは、2 個の 1 6 ビット入力ビット一致であるならば、1 を出力し、ビット一致でなければ、0 を出力する。L T ブロックは、アキュムレータから入力された上位 1 6 ビットが MX_2 によって M U から選択された 1 6 ビット値よりも小さい場合に、1 を出力する。比較は符号無しである。オペランドのビットパターンは組み合わせロジックがより簡単化されるように特に選択される。オペランドのビットパターンを利用するので、そのパターンをもう一度列挙する。

30

【1 4 1 3】

【表 1 4 9】

オペランド	CMD ₃₋₀
Init	0000
MLO	1110
MHI	1111

40

M H I 及び M L O は、ハイビットがセットされているので、I n i t のビットパターンと簡単に区別することができ、最下位ビットは M H I と M L O を区別するために使用できる。E E フラグ及び D E フラグは、R P L コマンドが発行されるたびに更新される。I n i

50

t ステージのため、2 個の値を設定し、M H I 及び M L O のため、E E 及び D E の値を適切に更新する。したがって、E E 及び D E のための W r i t e E n a b l e は以下の通りである。

【 1 4 1 4 】

【表 1 5 0 】

Logic ₁ :	Cycle AND (CMD ₇ = RPL)

3 2 ビットの A M T レジスタに関して、このレジスタには、R P L I n i t コマンドのときに、(M U から読み出された) A M の内容をロードし、R P L M L O コマンド及び R P L

M H I コマンドのために A M T レジスタを右へ 2 ビット位置シフトさせる。これは、R P L オペランドの最上位ビット (C M D ₃) に対して簡単にテストできる。したがって、A M T レジスタのための W r i t e E n a b l e 及び S h i f t E n a b l e は、以下の通りである。

【 1 4 1 5 】

【表 1 5 1 】

Logic ₂	Logic ₁ AND CMD ₃
Logic ₃	Logic ₁ AND -CMD ₃

L o g i c ₃ からの出力は、マルチプレクサ M X ₁ への入力としても有用である。なぜならば、それは、現在の 2 個のアクセスモードビット、又は 0 0 (これは、アクセスモード R W を表すので、D E 及び E E レジスタをリセットさせる) の何れかを通すために使用されるからである。したがって、M X ₁ は次の通りである。

【 1 4 1 6 】

【表 1 5 2 】

	出力	Logic ₃
M X ₁	A M T 出力	0
	00	1

R P L ロジックは、アキュムレータの上位 1 6 ビットだけを置換する。下位 1 6 ビットはそのまま通過する。しかし、(M [0 - 1 5] のうちの一つに対応する) M U からの 3 2 ビットのうちの、上位又は下位 1 6 ビットだけが使用される。したがって、M X ₂ は M H I と M L O を区別するために C M D ₀ をテストする。

【 1 4 1 7 】

【表 1 5 3 】

	出力	CMD ₀
M X ₂	下位 16 ビット	0
	上位 16 ビット	1

D E レジスタ及び E E レジスタを更新するロジックは、W R コマンドの擬似コードと一致する。A c c e s s M o d e 値が 0 0 (= R W、これは R P L

I N I T 中に出現する) である入力は、D E と E E の両方に 0 (正しい初期値) をロードさせることに注意する必要がある。E E は L o g i c ₄ からの結果がロードされ、D E は

Logic₅からの結果がロードされる。

【1418】

【表154】

Logic ₄	((AccessMode=MSR) AND EQ) OR (((AccessMode=NMSR) AND EE AND EQ))
Logic ₅	((AccessMode=MSR) AND LT) OR (((AccessMode=NMSR) AND DE) OR (((AccessMode=NMSR) AND EQ AND LT))

アキュムレータの上位16ビットはMへ書き込まれるべき値によって置き換えられる。したがって、Logic₆はWRコマンド擬似コードからのWEフラグと一致する。 10

【1419】

【表155】

Logic ₆	((AccessMode=RW) OR ((AccessMode=MSR) AND LT) OR ((AccessMode=NMSR AND (DE OR LT)))

Logic₆からの出力は、マルチプレクサMX₃による、アキュムレータからの元の16ビットと、M[0-15]からの値との間の選択を駆動するためそのまま使用される。アキュムレータからの16ビットが選択された場合（アキュムレータは変更されないで残される場合）、これは、アキュムレータ値がM[n]に書き込まれることを意味する。Mからの16ビット値が選択された場合（アキュムレータの上位16ビットを変更する場合）、これは、Mの16ビット値が変更されないことを意味する。したがって、MX₃は以下の形式をとる。 20

【1420】

【表156】

	出力	Logic ₆
MX ₃	MUからの16ビット	0
	Accからの16ビット	1

AMTをパリティチェックする意味はない。なぜならば、攻撃者はMX₃への入力を強制的に0にする方がよいからである（これにより、攻撃者は任意の値をMに書き込めるようになる）。しかし、攻撃者がチップ（全てタンパー検出テスト及び回路を含む）をレーザー切断する手順を選ぶならば、MX₃の入力を固定することにより制限選択テキスト攻撃を可能にさせるよりも優れたターゲットがある。 30

【1421】

ROR

図191は、ALUのRORブロックの概略ブロック図である。RORユニットは、ALU内のコンポーネントである。RORユニットは、認証チップのROR機能を実現するため設計される。RTMPという名前の1ビットレジスタは、RORユニットに格納される。RTMPはリセットで0にクリアされ、ROR 40

RBコマンド及びROR XRBコマンド中にセットされる。RTMPレジスタは、任意のタップ構造をもつ線形フィードバックシフトレジスタの実装を可能にさせる。XORブロックは、2個の1ビット入力、1ビット出力のXORである。RORnは、ブロックが便宜上示されているが、実際には、マルチプレクサMX₃に配線接続される。なぜならば、各ブロックは、単に、右へNビットシフトされた、32ビットの書き換えである。3個 50

全てのマルチプレクサ (MX_1 、 MX_2 及び MX_3) は 8 ビットの CMD 値に依存する。しかし、 ROR オペコードのビットパターンはロジック最適化の目的のために並べられる。オペランドのビットパターンを利用するので、このパターンを再度列挙する。

【1422】

【表157】

オペランド	CMD_{3-0}
InBit	0000
OutBit	0001
RB	0010
XRB	0011
IST	0100
ISW	0101
MTRZ	0110
1	0111
2	1001
27	1010
31	1100

$Logic_1$ は、 $RTMP$ に $Write Enable$ 信号を与えるために使用される。 $RTMP$ レジスタは、 ROR

RB 及び $ROR XOR$ コマンドの間だけ書き込まれるべきである。 $Logic_2$ は、 $InBit$ が使われたときならいつでも、制御信号を与えるために使用される。2 個の組み合わせロジックブロックは次の通りである。

【1423】

【表158】

$Logic_1$	Cycle AND ($CMD_{7-4} = ROR$) AND ($CMD_{3-1} = 001$)
$Logic_2$	Cycle AND ($CMD_{7-0} = ROR InBit$)

マルチプレクサ MX_1 を用いて、 $RTMP$ に格納されるビットを選択する。 $Logic_1$ は、予め RB と XRB の一方への CMD 入力の範囲を限定する。したがって、これらの二つを区別するため、 CMD_0 をテストするだけでよい。以下の表は、 CMD_0 と MX_1 から出力される値との間の関係を表現する。

【1424】

【表 1 5 9】

	出力	CMD ₀
MX ₁	Acc ₀	0
	XOR 出力	1

マルチプレクサMX₂を用いて、アキュムレータ入力のビット0を置換する予定の入力ビットを選択する。ここでは、少しの量の最適化を行う。なぜならば、個別の入力ビットは、典型的に、特定のオペランドに関係するからである。以下御表は、CMD₃₋₀とMX₂から出力される値との間の関係を表現する。

【1 4 2 5】

【表 1 6 0】

	出力	CMD ₃₋₀	コメント
MX ₂	Acc ₀	1xxx 又は 111	1,2,27,31
	RTMP	001x	RB,XRB
	InBit	000x	InBit,OutBit
	MU ₀	010x	IST,ISW
	MTRZ	110	MTRZ

最後のマルチプレクサMX₃は32ビット値の最終的な回転を行う。再度、CMDオペランドのビットパターンが利用される。

【1 4 2 6】

【表 1 6 1】

	出力	CMD ₃₋₀	コメント
MX ₃	ROR 1	0xxx	2,27,及び 31 以外の全て
	ROR 2	1xx1	2
	ROR 27	1x1x	27
	ROR 31	11xx	31

MinTicksユニット

図192は、MinTicksユニットのコンポーネント間のデータフロー及び関係を示す図である。MinTicksユニットは、認証チップ内の鍵付きオペレーションの間のプログラマブル最小遅延（カウントダウンを利用）の役目を担う。MinTicksユニットに収容されるロジック及びレジスタは、両方のタンパー検出ラインによってカバーする必要がある。これは、攻撃者が鍵付きの関数への呼び出しの間の時間を変更できないことを保証する。殆ど全てのMinTicksユニットは通常のCMOSで実現可能である。なぜならば、鍵は、殆どのこのユニットを通らないからである。しかし、アキュムレータは、SET

MTR命令で使用される。その結果として、この回路のこの小さい区域は、ノンフラッシングCMOSで実現しなければならない。MinTicksユニットの残りの部分はノンフラッシングCMOSで実現しなくてもよい。しかし、MTRZラッチ（以下を参照）はパリティチェックしなければならない。

【1 4 2 7】

MinTicksユニットは、MTR（MinTicks残量（MinTicksRem

a i n i n g)) という名前の 3 2 ビットレジスタを含む。M T R レジスタは、次の鍵付き関数を読み出せるまでの残りクロックチック数を格納する。各サイクルに、M T R の値は、その値が 0 になるまで 1 ずつ減少させられる。M T R が 0 に達すると、それ以上減少しない。M T R Z (M i n T i c k s 残量レジスタ零 M i n T i c k s R e g i s t e r Z e r o) という名前の補助的な 1 ビットレジスタは、M T R レジスタの現在の零性を反映する。M T R Z レジスタが 0 である場合、M T R Z は 1 であり、M T R Z レジスタが 0 ではない場合、M T R Z は 0 である。M T R レジスタはリセットによってクリアされ、S E T

M T R コマンドを用いて新しいカウントにセットされ、アキュムレータの現在値を M T R レジスタへ転送する。

【 1 4 2 8 】

ここで、

【 1 4 2 9 】

【表 1 6 2 】

Logic ₁	CMD = SET MTR

であり、且つ

【 1 4 3 0 】

【表 1 6 3 】

	出力	Logic ₁	MTRZ
MX ₁	Acc	1	-
	MTR-1	0	0
	0	0	1

である。

【 1 4 3 1 】

サイクルは M T R 及び M T R Z の W r i t e E n a b l e に接続されるので、これらのレジスタは実行サイクル中に、即ち、サイクル = 1 であるときに限り更新される。2 個の V A L ユニットは、タンパー防止及び検出ラインに接続された検証ユニットであり、各 V A L は O K ビットを含む。O K ビットはリセットで 1 にセットされ、サイクル毎に両方のタンパー検出ラインからの C h i p O K 値と O R 演算される。O K ビットはそのユニットを通過する各データビットと A N D 演算される。V A L₁ の場合、M T R からの実効出力は 0 であり、これは、デクリメンターユニットからの出力が全て 1 であることを示し、これにより、M T R Z は 0 のままであり、攻撃者が鍵付き関数を使用することは阻止される。また、V A L₁ は、M T R レジスタのパリティを検証する。パリティチェックが失敗した場合、消去タンパー検出ラインがトリガーされる。V A L₂ の場合、チップ改ざんされたならば、M T R Z からの実効出力は 0 であり、M i n T i c k s R e m a i n i n g レジスタは未だ 0 に到達していないことを示し、これにより、攻撃者が鍵付き関数を使用することは阻止される。

【 1 4 3 2 】

プログラムカウントユニット

図 1 9 2 は、プログラムカウントユニットのブロック図である。プログラムカウントユニット (P C U) は、9 ビットの P C (プログラムカウンタ) と、分岐及びサブルーチン制御用のロジックと、を含む。プログラムカウントユニットは、通常の C M O S で実現できる。なぜならば、鍵はこのユニットを通らないからである。このユニットはノンフラッシング C M O S で実現しなくてもよい。しかし、ラッチはパリティチェックされる。その上

、メモリユニットに収容されたロジック及びレジスタは、P Cが攻撃者によって変更できないことを保証するため、両方のタンパー検出ラインで覆う必要がある。P Cは実際には6レベル×9ビットのP C A（P Cアレイ）として実装され、3ビットのS P（スタックポインタ）レジスタによってインデックス付けされる。P C及びS Pレジスタはリセットで全てクリアされ、オペコードに従ってプログラム制御のフロー中に更新される。P Cの現在値は、サイクル0（フェッチサイクル）の間にM Uへ出力される。P Cは、実行されるコマンドに基づいてサイクル1（実行サイクル）の間に更新される。殆どのケースで、P Cは単に1ずつ増加する。しかし、（サブルーチン、又はその他のジャンプの形式により）分岐が現れたとき、P Cはあ多らしい値で置き換えられる。新しいP C値を計算するメカニズムは、処理されるオペコードに依存する。

10

【1433】

A D Dブロックはモジュロー 2^9 の単純なアダーである。入力は、P C値と、1（P Cを1ずつ増加するため）、又は9ビットオフセット（ハイビットがセットされ、下位8ビットはM Uから）の何れか一方である。“+1”ブロックは、3ビット入力を使用し、1ずつ増加する（ラップ付き）。“-1”ブロックは、3ビット入力を使用し、1ずつ減少する（ラップ付き）。種々のP C制御の形式は以下の通りである。

【1434】

【表164】

コマンド	動作
JSR, JSI (ACC)	PCの古い値を後のためスタックに保存。 新しいPCは9ビット値であり、ビット0=0であり(したがって、サブルーチン偶数アドレスから始まる)、アドレスの上位8ビットはM Uに由来する(MU 8ビット値はJSR用のジャンプテーブル1、及び、JST用のジャンプテーブル2である。)
JSI RTS	スタックからPCの古い値をポップし、新しいPCを得るため1ずつインクリメントする。
TBR	ZフラグがTRBテストにマッチするならば、PCを9ビット値で置換し、ビット0=0であり、上位8ビットはM Uに由来する。それ以外の場合、現在のPCを1ずつインクリメントする。
DBR C1, DBR C2	C1Z又はC2Z(DBR C1用のC1Z、DBR C2用のC2Z)がセットされている場合に限り、9ビットオフセット(MUからの8ビットと、ハイビット=1)を現在のPCに加える。それ以外の場合、現在のPCを1ずつインクリメントする。
その他の全て	現在のPCを1ずつインクリメントする。

20

30

J S RとJ S I（A C C）に対して同じ動作がとられるので、L o g i c₁でそのケースを具体的に検出する。同じ考え方によって、L o g i c₂でJ S I R T Sを具体的にテストする。

【1435】

【表165】

Logic ₁	(CMD _{7:5} = 001) OR (CMD _{7:3} = 01001)
Logic ₂	CMD _{7:3} = 01000

40

P Cを更新するとき、P Cが完全に新しい項目によって置き換えられるべきか、又はアダーの結果によって置き換えられるべきかを決定しなければならない。これは、J S R及びJ S I（A C C）の場合に成り立つが、テストビットがアキュムレータの状態と一致する限り、T B Rにも成り立つ。T B R以外の全ては、L o g i c₁によってテストされるので、L o g i c₃はL o g i c₁の出力を入力として含む。L o g i c₃からの出力は、次に、新しいP C値を得るため、マルチプレクサM X₂によって使用される。

【1436】

【表 1 6 6】

Logic ₂	Logic ₁ OR ((CMD _{7,4} = TBR) AND (CMD ₃ XOR Z))

【 1 4 3 7】

【表 1 6 7】

	出力	Logic ₃
MX ₂	アダーからの出力	0
	置換値	1

10

9ビットアダーへの入力は、1ずつ増加するか（普通のケース）、又はMUから読み出されたオフセットを加算するか（DBRコマンド）に依存する。Logic₄はtestを生成する。Logic₄からの出力は、これに応じてマルチプレクサMX₃によってそのまま使用される。

【 1 4 3 8】

【表 1 6 8】

Logic ₄	((CMD _{7,3} = DBR C1) AND C1Z) OR (CMD _{7,3} = DBR C2) AND C2Z))

20

【 1 4 3 9】

【表 1 6 9】

	出力	Logic ₄
MX ₃	アダーからの出力	0
	置換値	1

30

最後に、どのPCエントリーを使用するかについての選択は、SPの現在値に依存する。サブルーチンに入るとき、SPインデックス値はインクリメントされ、サブルーチンから戻るとき、SPインデックス値はデクリメントされる。全てのケースで、且つコマンドをフェッチしたいとき（サイクル0）、SPの現在値を使用する。Logic₁は、サブルーチンに入るときを通知し、Logic₂はサブルーチンから戻るときを通知する。マルチプレクサ選択は、したがって、以下のように定義される。

【 1 4 4 0】

【表 1 7 0】

	出力	Cycle/Logic ₁ /Logic ₂
MX ₁	SP-1	1x1
	SP+1	11x
	SP	0xx OR 00

2個のVALユニットは、タンパー防止及び検出ラインに接続された検証ユニットであり、各VALはOKビットを含む。OKビットはリセットで1にセットされ、サイクル毎に両方のタンパー検出ラインからのChip OK値とOR演算される。OKビットはそのユ

50

ニットを通過する各データビットとAND演算される。両方のVALユニットは、データビットをパリティチェックし、それらが有効であることを保証する。パリティチェックが失敗した場合、消去タンパー検出ラインがトリガーされる。VAL₁の場合、SPレジスタからの実効出力は、チップが改ざんされたならば、常に0である。これにより、攻撃者がサブルーチンを実行することは阻止される。VAL₂の場合、実効PC出力は、チップが改ざんされたならば、常に0である。これにより、攻撃者がプログラムコードを実行することは阻止される。

【1441】

メモリユニット

メモリユニット(MU)は、認証チップの内部メモリを含む。内部メモリは、9ビットのアドレスによってアドレス指定され、9ビットのアドレスはアドレス生成器ユニットから渡される。メモリユニットは、アドレスに応じて、適切な32ビット及び8ビット値を出力する。また、メモリユニットは、特殊なプログラミングモードを担い、これにより、プログラムがフラッシュメモリに入力される。メモリユニット全体の内容は、タンパーリングから保護されるべきである。したがって、メモリユニットに収容されたロジック及びレジスタは、両方のタンパー検出ラインによって覆われるべきである。これは、プログラムコード、鍵、及び中間データ値が攻撃者によって変更できないことを保証する。全てのフラッシュメモリは、マルチステートであり、無効電圧で読み出されたときにチェックしなければならない。32ビットRAMもパリティチェックされるべきである。メモリユニットを通る32ビットデータパスは、ノンフラッシングCMOSで実現できる。なぜならば、鍵はそのデータパスに沿って送られるからである。8ビットデータパスは、通常のCMOSで実現できる。なぜならば、鍵はそのデータパスに沿って送られないからである。

【1442】

定数

定数メモリ領域のアドレスレンジは、0000000000-0000001111である。したがって、レンジは00000xxxxである。しかし、次の48アドレスが予備であるならば、これは復号化中に利用できる。定数メモリ領域は、アドレスの上位3ビット(Adr₈₋₆=000)によって選択され、下位4ビットは組み合わせロジックへ供給され、その4ビットは、以下の通り、32ビット出力値にマッピングされる。

【1443】

10

20

30

【表 1 7 1】

Adr ₃₋₀	出力値
0000	0x00000000
0001	0x36363636
0010	0x5C5C5C5C
0011	0xFFFFFFFF
0100	0x5A827999
0101	0x6ED9EBA1
0110	0x8F1BBCDC
0111	0xCA62C1D6
1000	0x67452301
1001	0xEFCDAB89
1010	0x98BADCFE
1011	0x10325476
11xx	0xC3D2E1F0

10

20

30

R A M

3 2 エントリーの 3 2 ビット R A M のアドレス空間は、0 0 1 0 0 0 0 0 0 – 0 0 1 0 1 1 1 1 である。したがって、レンジは 0 0 1 0 x x x x x である。R A M メモリ領域は、したがって、アドレスの上位 4 ビット (Adr₈₋₅ = 0 0 1 0) によって選択され、下位 5 ビットはアドレス指定する値を選択する。連続的な 3 2 エントリーのアドレス空間が与えられた場合、R A M は、単純な 3 2 × 3 2 ビット R A M として簡単に実現できる。C P U は、特殊な方法でレンジ 0 0 0 0 0 – 1 1 1 1 1 からの各アドレスを取り扱うが、R A M アドレスデコーダ自体は、アドレスを特別に取り扱うわけではない。全ての R A M 値はリセットで 0 にクリアされるが、プログラムコードはこのことを当てにしている。

40

【1 4 4 4】

フラッシュメモリ変数

3 2 ビット幅フラッシュメモリのアドレス空間は、0 0 1 1 0 0 0 0 0 – 0 0 1 1 1 1 1 1 である。したがって、レンジは 0 0 1 1 x x x x x である。フラッシュメモリ領域は、したがって、アドレスの上位 4 ビット (Adr₈₋₅ = 0 1 1 1) によって選択され、下位 5 ビットはアドレス指定する値を選択する。フラッシュメモリは特別な消去が必要である。フラッシュメモリの消去を完了するためにはかなり長い時間を要する。したがって、

50

Wait 信号は、CLR コマンドの受信時にフラッシュコントローラの内部にセットされ、要求されたメモリが消去されたときに限りクリアされる。内部的に、特定のメモリレンジの消去ラインは一つに結合され、その結果として、以下の表に示されるように、2 ビットだけが要求される。

【1445】

【表172】

Adr _{4:3}	消去レンジ
00	R ₀₋₄
01	MT,AM,K1 ₀₋₄ ,K2 ₀₋₄
10	個別の M アドレス (Adr)
11	IST,ISW

フラッシュ値はリセットによって変更されないが、プログラムコードは、ガーベッジ以外では、（製造後に）フラッシュの初期値をとるべきではない。フラッシュアドレスを利用するオペレーションは、LD、ST、ADD、RPL、ROR、CLR 及び SET である。全てのケースで、オペランド及びメモリ配置は、アドレス生成及び復号化を最小限に抑えるため、密接に連結される。フラッシュメモリの全変数区域は、プログラミングモードへ入ったとき、及び明確な物理的攻撃を検出したときに消去される。

【1446】

フラッシュメモリープログラム

384 エントリーの 8 ビット幅プログラムフラッシュメモリのアドレスレンジは、010000000-11111111 である。したがって、レンジは 01xxxxxxx-11xxxxxxx である。ROM スタートアドレスとアドレスレンジが与えられるならば、復号化は簡単である。CPU はアドレスレンジの一部を特殊な方法で取り扱うが、アドレスデコード自体はアドレスを特別に取り扱うわけではない。フラッシュ値はリセットによって変わらず、プログラムモードへ入ることによってのみクリアされる。製造後、フラッシュ内容はガーベッジであると思なされるべきである。384 バイトは、プログラミングモードにあるときに限り、状態機械によってロードされる。

【1447】

MU のブロック図

図193は、メモリユニットのブロック図である。図示されたロジックは、32 ビットデータ及び 8 ビットデータが別々のコマンドによって要求され、その結果として、復号化のために必要なビットが少なくなるということを利用する。図示されるように、32 ビット出力及び 8 ビット出力は常に生成される。認証チップの残りの適切なコンポーネントは、単に、実行されるコマンドに依存して、32 ビット値又は 8 ビット値を使用する。マルチプレクサ MX₁ は、真理表定数、RAM、及びフラッシュメモリの選択から出力された 32 ビットを選択する。3 出力の間で選択するためには、2 ビットだけが、即ち、Adr₆ 及び Adr₅ が要求される。このようにして、MX₂ の形式は以下の通りである。

【1448】

【表 1 7 3】

	出力	Adr _{6,5}
MX ₂	32 ビット真理表からの出力	00
	32 ビットフラッシュメモリからの出力	10
	32 ビット RAM からの出力	11

32 ビットフラッシュメモリの特定の部分を消去するロジックは Logic₁ によって満たされる。消去部分制御信号は、サイクル = 1 の間に、メモリの正しい部分への CLR コマンド中に限りセットされるべきである。単一の CLR コマンドがあるレンジのフラッシュメモリをクリアすることに注意する必要がある。Adr₆ は CLR のためのアドレスレンジとして十分である。なぜならば、レンジは、有効オペランドに対して常にフラッシュの範囲内に治まり、有効ではないオペランドに対して 0 である。32 ビット幅フラッシュメモリのレンジ全体は、（攻撃者によって、又は意図的にプログラミングモードに入ることによって）消去検出ラインがトリガーされたときに消去される。

【1 4 4 9】

【表 1 7 4】

Logic ₁	Cycle AND (CMD _{7,4} = CLR) AND Adr ₆

フラッシュメモリの特定の部分に書き込むためのロジックは、Logic₂ によって満たされる。Write Enable 制御信号は、サイクル = 1 の間に、フラッシュメモリレンジへの適切な ST コマンド中に限りセットされるべきである。Adr₆₋₅ だけのテストが許容される。なぜならば、ST コマンドだけがフラッシュまたは RAM へ有効に書き込むからである（Adr₆₋₅ が 00 であるならば、K2MX は 0 でなければならない。）。

【1 4 5 0】

【表 1 7 5】

Logic ₂	Cycle AND (CMD _{7,4} = ST) AND (Adr _{6,5} = 10)

WE (Write Enable) フラグは、SET WE コマンドと CLR WE コマンドの実行中にセットされる。Logic₃ はこれらの二つのケースをテストする。WE に書き込まれる実際のビットは CMD₄ である。

【1 4 5 1】

【表 1 7 6】

Logic ₃	Cycle AND (CMD _{7,5} = 001) AND (CMD _{3,0} = 0000)

メモリの RAM 領域へ書き込むためのロジックは、Logic₄ によって満たされる。Write Enable 制御信号は、サイクル = 1 の間に、RAM メモリレンジへの適切な ST コマンド中に限りセットされる。しかし、これは、X[N] への書き込みが許可されるかどうかを管理する WE フラグによって改ざんされる。X[N] レンジは RAM の上位半分であり、これは、Adr₄ を使用してテストできる。RAM のフルアドレスレンジとして Adr₆₋₅ だけをテストすることが容認できる。なぜならば、ST コマンドはフラッシュ又は RAM だけに書き込むからである。

【1 4 5 2】

【表 1 7 7】

Logic ₄	Cycle AND (CMD _{7:4} = ST) AND (Adr _{6:5} = 11) AND ((Adr ₄ AND WE) OR (-Adr ₄))

3 個の V A L ユニットは、タンパー防止及び検出ラインに接続された検証ユニットであり、各 V A L は O K ビットを含む。O K ビットはリセットで 1 にセットされ、サイクル毎に両方のタンパー防止及び検出ラインからの C h i p O K 値と O R 演算される。O K ビットはそのユニットを通過する各データビットと A N D 演算される。V A L ユニットは、データビットをパリティチェックし、それらが有効であることを保証する。V A L₁ 及び V A L₂ は、各データビットの状態をチェックすることにより検証し、V A L₃ はパリティチェックを実行する。妥当性テストが失敗したとき、消去タンパー検出ラインがトリガーされる。V A L₁ の場合、プログラムフラッシュからの実効出力は、チップが改ざんされているならば、常に 0 である (T B R

0 として解釈される)。これは攻撃者が有用な命令を実行することを阻止する。V A L₂ の場合、実効 3 2 ビットは、チップが改ざんされたならば、常に 0 である。これにより、攻撃者は、鍵、又は中間記憶値を入手できない。8 ビットフラッシュメモリは、プログラムコード、ジャンプテーブル、及びその他のプログラム情報を保持するため使用される。

3 8 4 バイトのプログラムフラッシュメモリは、(アドレスレンジ 0 1 x x x x x x x - 1 1 x x x x x x x を使用して) 完全な 9 ビットのアドレスにより選択される。プログラムフラッシュメモリは、(攻撃者によって、又はプログラミングモード検出ユニットが原因でプログラミングモードに入ることによって) 消去検出ラインがトリガーされたときに限り消去される。消去検出ラインがトリガーされたとき、プログラムフラッシュメモリユニット内の小型状態機械は、8 ビットフラッシュメモリを消去し、消去を検証し、シリアル入力から新しい内容 (3 8 4 バイト) にロードする。以下の擬似コードは、消去検出ラインがトリガーされたときに実効される状態機械ロジックを説明する：

Set WAIT output bit to prevent the remainder of the chip from functioning

Fix 8-bit output to be 0

Erase all 8-bit Flash memory

Temp ← 0

For Adr = 0 to 383

Temp ← Temp OR Flash_{Adr}

IF (Temp ≠ 0)

Hang

For Adr = 0 to 383

Do 8 times

Wait for InBitValid to be set

ShiftRight[Temp, InBit]

Set InBitUsed control signal

Flash_{Adr} ← Temp

Hang

プログラミングモード状態機械実行中に、0 が 8 ビット出力にセットされる。0 コマンド

は、認証チップの残りの部分に、コマンドを T B R 0 として解釈させる。チップが全部で 3 8 4 バイトをプログラムフラッシュメモリへ読み込んだとき、チップはハングする（無限ループに入る）。次に、認証チップはリセットされ、プログラムは正常に使用される。消去は、8 ビットプログラムフラッシュメモリの新しい内容をロードするために使用される 8 ビットレジスタによって検証されることに注意すべきである。これは、攻撃が成功する見込みを低下させるために役立つ。なぜならば、プログラムコードは、消去を認証するため使用されるレジスタが攻撃者によって破壊された場合に、適切にロードされ得ないからである。更に、状態機械全体は、両方のタンパー検出ラインによって保護される。

【1 4 5 3】

アドレス生成器ユニット

アドレス生成器ユニットは、メモリユニット（M U）へアクセスするための実効アドレスを生成する。サイクル 0 において、P C は次のオペコードをフェッチするため M U へ渡される。アドレス生成器は、サイクル 1 の実効アドレスを生成するため、戻されたオペコードを解釈する。サイクル 1 において、生成されるアドレスは M U へ渡される。アドレス生成器ユニットに格納されたロジック及びレジスタは、両方のタンパー検出ラインによってカバーされる。これは、攻撃者が生成されたアドレスを変更し得ないことを保証する。殆ど全部のアドレス生成器ユニットは通常の C M O S で実現できる。なぜならば、鍵が殆どこのユニットを通らないからである。しかし、アキュムレータの 5 ビットは J S I アドレス生成で使用される。この結果として、回路のこの小さい区域は、ノンフラッシング C M O S で実現されてもよい。しかし、カウンタ及び計算されたアドレスのラッチはパリティチェックされるべきである。タンパー検出ラインの何れかが破断しているならば、アドレス生成器ユニットは、各サイクルでアドレス 0 を生成し、全てのカウンタは 0 に確定される。これは、攻撃者がリセットを禁止状態にするか、及び／又は、消去回路を禁止状態にした場合に始めて効果がある。なぜならば、正常な状況下では、タンパー検出ラインの破断は、リセット又は全てのフラッシュメモリの消去を生じさせるからである。

【1 4 5 4】

アドレス生成の背景

アドレス生成用ロジックは、種々のオペコードとオペランドの組み合わせを調べることが必要である。オペコード／オペランドと、アドレスとの関係はこのセクションで検討され、アドレス生成器ユニットの基礎として使用される。

【1 4 5 5】

定数

下位 4 エントリーは、汎用用途、並びに、H M A C アルゴリズムのための簡単な定数である。L D K オペランドの下位 4 ビットは、これらの 4 個の値、即ち、0 0 0 0、0 0 0 1、0 0 1 0 及び 0 0 1 1 のメモリ内アドレスの下位 3 ビットにそのまま対応する。y 定数及び h 定数も L D K コマンドによってアドレス指定される。しかし、アドレスは、オペランドの下位 3 ビットを C 1 カウンタ値のインバースで O R 演算し、そのままのオペランドの第 4 ビットを保持することによって生成される。このようにして、L D K y の場合、y オペランドは 0 1 0 0 であり、L D K h の場合、h オペランドは 1 0 0 0 である。反転した C 1 値は、y に対して 0 0 0 - 0 1 1 の範囲にあり、h に対して、0 0 0 - 1 0 0 の範囲にあり、O R 演算結果は正確なアドレスを与える。全ての定数に対して、最終アドレスの上位 5 ビットは常に 0 0 0 0 0 である。

【1 4 5 6】

R A M

変数 A - T は、それらのオペランド値の下位 3 ビットに直接関係したアドレスをもつ。即ち、L D、S T、A D D、L O G、及び X O R コマンドのオペランド値 0 0 0 0 - 0 1 0 1 の場合、並びに、L O G コマンドのオペランド値 1 0 0 0 - 1 1 0 1 の場合、下位 3 ビットのオペランドアドレスビットは、最終アドレスを生成するため、0 0 1 0 0 0 の一定

上位6ビットアドレスと共に使用される。残りのレジスタ値は、インデックス付きメカニズムによってアクセスできる。変数A-E、B160及びHは、C1カウンタ値によるインデックス付きとしてのみアクセス可能であり、Xは、N₁、N₂、N₃及びN₄によってインデックス付けされる。LD、ST及びADDコマンドの場合、C1によってインデックス付けされたAEのアドレスは、オペランドの下位3ビット(000)を取り、C1カウンタ値とOR演算することにより生成される。しかし、H及びB160のアドレスは、同じ方法では生成できない(そうでなければ、RAMアドレス空間は不連続になるであろう)。したがって、簡単な組み合わせロジックは、AEを0000に、Hを0110に、B160を1011に変換する。最終アドレスは、C1を4ビット値に加算し(4ビット値が得られる)、00100の一定上位5ビットアドレスを前に付加することによって得られる。最後に、レジスタのXレンジは、N₁、N₂、N₃及びN₄によるインデックス付きとしてのみアクセス可能である。XORコマンドの場合、インデックス付けのため任意のN₁-₄を使用可能であり、一方、LD、ST及びADDの場合、N₄だけが使用可能である。LD、ST及びADDにおけるXのオペランドは、X_{N₄}オペランドと同じであるため、オペランドの下位2ビットは、使用すべきNを選択する。アドレスは、このようにして、00101の一定上位5ビット値と、選択されたNカウンタに由来する下位4ビットとして生成される。

【1457】

フラッシュメモリー変数

変数MT及びAMのアドレスは、関連したコマンドのオペランドから生成される。オペランドの4ビットはそのまま使用され(0110及び0111)、00110の一定上位5ビットアドレスが前に付加される。変数R₁-₅、K1₁-₅、K2₁-₅、及びM₀-₇は、C1カウンタ値のインバースによるインデックス付きとしてのみアクセス可能である(付加的には、Rの場合には実際のC1値による)。簡単な組み合わせロジックは、R及びRFを00000に、K1又はK2がアドレス指定されているかどうかに応じてKを01000又は11000に、M(MHI及びMLOを含む)を10000に変換する。最終アドレスは、C1を5ビット値とOR演算(又は加算)し(RFの場合には、C1をそのまま使用し)、0011の一定上位4ビットを前に付加することによって獲得される。変数IST及びISWは各々が1ビット値に過ぎないが、任意のビット数によって実装できる。データは、0x00000000又は0xFFFFFFFの何れかとして読み書きされる。それらは、ROR、CLR及びSETコマンドだけによってアドレス指定される。RORの場合、オペランドの下位ビットは、00111111の一定上位8ビット値と合成され、IST及びISWに対して、それぞれ、0011111110及び0011111111が得られる。なぜならば、他のRORオペランドはメモリを使用せず、IST及びISW以外の場合には、返された値は無視できるからである。SET及びCLRの場合、IST及びISWは、0011の一定上位4ビットをIST(0100)から11110へのマッピング、及びISW(0101)から11111へのマッピングと組み合わせることによりアドレス指定される。IST及びISWは、RAMからのE及びTと同じオペランド値を共有するので、同じ復号化ロジックを下位5ビットのため使用することが可能である。最終アドレスは、ビット4、3及び1がセットされることを要求する(これは、オペランド値010xのテストの結果にOR演算することより行われる)。

【1458】

フラッシュメモリープログラム

プログラムフラッシュメモリ内でルックアップするためのアドレスは9ビットPC(サイクル0の場合)又は9ビットAdrレジスタ(サイクル1の場合)から直接もたらされる。TBR、DBR、JSR及びJSIのようなコマンドは、プログラムメモリ内の特定のアドレスでテーブルに記憶されたデータにしたがってPCを変更する。その結果として、アドレス生成は、一部の定数アドレスコンポーネントを利用し、コマンドオペランド(又はアキュムレータ)は実効アドレスの下位ビットを形成する。

【1459】

10

20

30

40

50

【表 1 7 8】

コマンド	アドレスレンジ	アドレスの一定(上位)部分	アドレスの可変(下位)部分
TBR	010000xxx	010000	CMD ₂₋₀
JSR	0100xxxxx	0100	CMD ₄₋₀
JSI ACC	0101xxxxx	0101	Acc ₄₋₀
DBR	011000xxx	011000	CMD ₂₋₀

アドレス生成器ユニットのブロック図

図 1 9 4 はアドレス生成器ユニットの概略ブロック図である。アドレス生成器ユニットからの一次出力は、以下の表に示されるように、マルチプレクサMX₁によって選択される。

【1 4 6 0】

【表 1 7 9】

	出力	Cycle
MX ₁	PC	0
	Adr	1

CMDデータとMUからの8ビットを区別することが重要である：

サイクル0では、8ビットデータラインは、後のサイクル1で実行されるべき次の命令を保持する。この8ビットコマンド値は実効アドレスをデコードするため使用される。これに対して、CMDの8ビットデータは前の命令を保持するので、無視されるべきである；サイクル1では、CMDラインは現在実行中の命令（サイクル0中に8ビットデータラインに保持されていた命令）を保持し、一方、8ビットデータラインは命令からの実効アドレスのデータを保持する。CMDデータはサイクル1中に実行されなければならない。

【1 4 6 1】

したがって、MU又はCMDからの9ビットデータの選択は、以下の表に示されるようにマルチプレクサMX₃によって行われる。

【1 4 6 2】

【表 1 8 0】

	出力	Cycle
MX ₃	MUからの8ビットデータ	0
	CMD	1

9ビットAdrレジスタはサイクル0毎に更新されるので、AdrのWrite Enableは～サイクルに接続される。カウンタユニットはカウンタC1、C2（内部的に使用される）と、選択されたNインデックスを生成する。その上、カウンタユニットは、プログラムカウンタユニットによって使用されるフラグC1Z及びC2Zを出力する。様々な*GENユニットは、サイクル0中に、特有のコマンドタイプのアドレスを生成し、マルチプレクサMX₂は、PC（即ち、8ビットデータライン）によってプログラムメモリから読み出されるようなコマンドに基づいてそれらの間で選択を行う。生成された値は以下の通り

である。

【1463】

【表181】

ブロック	アドレスが生成された目的のコマンド
JSIGEN	JSI ACC
JSRGEN	JSR,TBR
DBRGEN	DBR
LDKGEN	LDK
RPLGEN	RPL
VARGEN	LD,ST,ADD,LOG,XOR
BITGEN	ROR,SET
CLRGEN	CLR

10

マルチプレクサMX₂は以下の選択規準をもっている。

【1464】

【表182】

	出力	MUからの8ビットデータ値
MX ₂	JSIGENからの9ビット値	01001xxx
	JSRGENからの9ビット値	001xxxxx OR 0000xxxx
	DBRGENからの9ビット値	0001xxxx
	LDKGENからの9ビット値	1110xxxx
	RPLGENからの9ビット値	1101xxxx
	VARGENからの9ビット値	10xxxxxx OR 1x11xxxx
	BITGENからの9ビット値	0111xxxx OR 1100xxxx
	CLRGENからの9ビット値	0110xxxx

20

VAL₁ユニットは、タンパー防止及び検出回路に接続された検証ユニットである。それはOKビットを含み、OKビットはリセットで1にセットされ、サイクル毎に両方のタンパー防止及び検出ラインからのChipOK値とOR演算される。OKビットは、使用可能前の9ビットの実効アドレスとAND演算される。チップが改ざんされているならば、アドレス出力は常に0であり、これにより、攻撃者はメモリの他の部分へのアクセスが妨げられる。VAL₁ユニットは、実効アドレスビットにパリティチェックを実行し、それらが改ざんされていないことを保証する。パリティチェックが失敗であるならば、消去タンパー検出ラインがトリガーされる。

30

【1465】

JSIGEN

図195はJSIGENユニットの概略ブロック図である。JSIGENユニットはJSI

40

I ACC命令のためのアドレスを生成する。実効アドレスは、単に、JSIテーブル(0101)用のアドレスの4ビット上位部分と、アキュムレータ値の下位5ビットと、の連結である。

【1466】

アキュムレータは他の時点(ジャンプアドレスが生成されないとき)で鍵を保持することがあるので、その値は秘匿されなければならない。その結果として、このユニットはノンフラッシングCMOSで実現される。マルチプレクサMX₁は、単に、コマンドがJSIGENであるかどうかに基づいて、アキュムレータからの下位5ビット又は0の間で選択

50

を行う。マルチプレクサ MX_1 は以下の選択規準をもっている。

【1467】

【表183】

	出力	CMD_{7-0}
MX_1	アキュムレータ ₄₋₀	JSI ACC
	00000	-(JSI ACC)

10

JSRGEN

図196はJSRGENユニットの概略ブロック図である。JSRGENユニットはJSR及びTBR命令のためのアドレスを生成する。実効アドレスは、単に、JSRテーブル(0100)用のアドレスの4ビット上位部分と、オペランドからのテーブル内のオフセット(JSRコマンド用の3ビットと、TBR用の3ビット+定数0ビット)と、の連結からもたらされる。ここで、 $Logic_1$ は実効アドレスのビット3を生成する。このビットはJSRの場合にはビット3であり、TBRの場合には0である。

【1468】

【表184】

20

$Logic_1$	$bit_5 \text{ AND } bit_3$

JSR命令はビット5に1があるので(これに対して、TBRはこのビットが0である)、これをビット3とAND演算することにより、JSRの場合にはビット3を生成し、TBRの場合には0を生成する。

【1469】

DBRGEN

図197は、DBRGENユニットの概略ブロック図である。DBRGENユニットはDBR命令のためのアドレスを生成する。実効アドレスは、DBRテーブル(011000)用のアドレスの6ビット上位部分と、オペランドの下位3ビットと、の連結からもたらされる。

【1470】

LDKGEN

図198は、LDKGENユニットの概略ブロック図である。LDKGENユニットはLDK命令のためのアドレスを生成する。実効アドレスは、LDKテーブル(00000)用のアドレスの5ビット上位部分と、オペランドのハイビットと、オペランドの下位3ビット(下位が定数の場合)、又はC1とOR演算されたオペランドの下位3ビット(インデックス付き定数の場合)との連結からもたらされる。

【1471】

OR₂ブロックは、C1の3ビットを、MUからの8ビットデータ出力からの下位3ビットとOR演算するだけである。マルチプレクサ MX_1 は、オペランドの上位ビットがセットされているか、又はセットされていないかに基づいて、実際のデータビットと、C1がOR演算されたデータビットとの間で選択を行うだけである。マルチプレクサへのセレクト入力は、簡単なORゲートであり、 bit_2 を bit_3 とOR演算する。マルチプレクサ MX_1 は以下の選択規準をもっている。

【1472】

50

【表 1 8 5】

	出力	bit ₃ 又は bit ₂
MX ₁	Bit ₂₋₀	0
	OR ブロックからの出力	1

R P L G E N

図 1 9 9 は、R P L G E N ユニットの概略ブロック図である。R P L G E N ユニットは R P L 命令のためのアドレスを生成する。K 2 M X が 0 であるとき、実効アドレスは定数 0 0 0 0 0 0 0 0 0 である。K 2 M X が 1 であるとき（M からの読み出しが有効な値を返すことを示す）、実効アドレスは、M（0 0 1 1 1 0）用のアドレスの 6 ビット上位部分と、C 1 の現在値の 3 ビットと、の連結からもたらされる。

【1 4 7 3】

マルチプレクサ M X₁ は、K 2 M X の現在値に基づいて、2 個のアドレスの間で選択を行う。したがって、マルチプレクサ M X₁ は以下の選択規準をもっている。

【1 4 7 4】

【表 1 8 6】

	出力	K2MX
MX ₁	000000000	0
	001110 C1	1

V A R G E N

図 2 0 0 は、V A R G E N ユニットの概略ブロック図である。V A R G E N ユニットは L D、S T、A D D、L O G 及び X O R 命令のためのアドレスを生成する。K 2 M X 1 ビットフラグは、M からの読み出しが定数 0 アドレス（0 を返すが、書き込み不可である）にマッピングされるか、オペランドが K を指定したとき、K 1 と K 2 のどちらがアクセスされるか、を決定するため使用される。4 ビットアダーブロックは 2 組の 4 ビット入力を取り、モジュロ 2⁴ の加法で、4 ビットの出力を生成する。シングルビットレジスタ K 2 M X は、C L R K 2 M X 命令、又は S E T K 2 M X 命令の実行中だけに書き込まれる。L o g i c₁ は、これらの条件に基づいて K 2 M X W r i t e E n a b l e をセットする。

【1 4 7 5】

【表 1 8 7】

Logic ₁	Cycle AND bit ₇₋₀ = 011x0001

S E T 命令中に K 2 M X 変数に書き込まれるビットは 1 であり、C L R 命令中は 0 である。入力ビットのソースとして、オペコードの下位ビット（b i t₄）を使用するのが好都合である。アドレス生成中に、組み合わせロジックとして実装された真理表は、以下の通り、ベースアドレスの一部を決定する。

【1 4 7 6】

10

20

30

40

【表 1 8 8】

bit _{7,4}	bit _{3,0}	説明	出力値
LOG	x	A,B,C,D,E,T,MT,AM	00000
≠ LOG	0xxx OR 1x00	A,B,C,D,E,T,MT,AM, AE[C1], R[C1]	00000
≠ LOG	1001	B160	01011
≠ LOG	1010	H	00110
≠ LOG	111x	X,M	10000
≠ LOG	1101	K	K2MK 1000

10

真理表は5ビットの出力を生成するが、下位4ビットは4ビットアダーへ送られ、そこで、下位4ビットはインデックス値（C1、N又はオペランド自体の下位3ビット）に加算される。最上位ビットはアダーへ送られ、5ビット結果を生成するために、アダー結果からの4ビット結果の前に付加される。アダーへの第2の入力はマルチプレクサMX₁に由来し、マルチプレクサMX₁は、C1、N、及びオペランド自体の下位3ビットからインデックス値を選択する。C1は3ビットしかないが、4番目のビットは定数0である。マルチプレクサMX₁は以下の選択規準をもっている。

【1 4 7 7】

【表 1 8 9】

20

	出力	bit _{7,0}
MX ₁	データ _{2,0}	(bit ₃ = 0) OR (bit _{7,4} = LOG)
	C1	((bit ₃ = 1) AND (bit _{2,0} ≠ 111) AND ((bit _{7,4} = 1x11) OR (bit _{7,4} = ADD)))
	N	((bit ₃ = 1) AND (bit _{7,4} = XOR)) OR (((bit _{7,4} = 1x11) OR (bit _{7,4} = ADD)) AND (bit _{3,0} = 1111))

実効アドレスの第6ビット（bit₅）は、RAMアドレスの場合に0であり、フラッシュメモリアドレスの場合に1である。フラッシュメモリアドレスは、MT、AM、R、K、及びMである。bit₅の計算はLogic₂によって与えられる。

30

【1 4 7 8】

【表 1 9 0】

Logic ₂	((bit _{3,0} = 110) OR (bit _{3,0} = 011x) OR (bit _{3,0} = 110x)) AND ((bit _{7,4} = 1x11) OR (bit _{7,4} = ADD))

定数1ビットが表現され、実効アドレスは全部で7ビットになる。これらのビットは、K2MXが0でなく、命令がLD、ADD又はST

M[C1]ではない限り、実効アドレスを形成する。後者では、実効アドレスは定数アドレス0000000である。両方のケースで、2個の0ビットが最終的な9ビットアドレスを形成するため前に追加される。この計算は、以下に示され、Logic₃及びマルチプレクサMX₂によって行われる。

40

【1 4 7 9】

【表 1 9 1】

Logic ₃	-K2MX AND (bit _{3,0} = 1110) AND ((bit _{7,4} = 1x11) OR (bit _{7,4} = ADD))

【1 4 8 0】

【表 1 9 2】

	出力	Logic ₃
MX ₂	計算されたビット	0
	0000000	1

C L R G E N

図 2 0 1 は、C L R G E N ユニットの概略ブロック図である。C L R G E N ユニットは C L R 命令のためのアドレスを生成する。実効アドレスは常に、有効メモリアクセスオペランド用のフラッシュメモリにあり、無効オペランドに対して 0 である。C L R M [C 1] 命令は、(V A R G E N ユニットに保持された) K 2 M X フラグの状態とは無関係に、常に、M [C 1] を消去する。真理表は、単純な組み合わせロジックであり、以下の関係を実現する。

【1 4 8 1】

【表 1 9 3】

入力値 (bit ₃₋₀)	出力値
1100	00 1100 000
1101	00 1101 000
1110	00 1110 C1
1111	00 1111 110
-(11xx)	00 0000 000

真理表に必要なロジックを削減することは簡単である。なぜならば、全部で 4 種類の主要なケースにおいて、実効アドレスの最初の 6 ビットは 0 0 であり、その後にオペランド (b i t _{3 - 0}) が続く。

【1 4 8 2】

B I T G E N

図 2 0 2 は、B I T G E N ユニットの概略ブロック図である。B I T G E N ユニットは R O R 及び S E T 命令のためのアドレスを生成する。実効アドレスは常に、有効メモリアクセスオペランド用のフラッシュメモリにあり、無効オペランドに対して 0 である。R O R 及び S E T 命令は、I S T 及び I S W フラッシュメモリアドレスだけにアクセスし (オペランドの残りはレジスタにアクセスする)、簡単な組み合わせロジック真理表はアドレス生成に十分である。

【1 4 8 3】

【表 1 9 4】

入力値 (bit ₃₋₀)	出力値
010x	00111111 bit ₀
-(010x)	000000000

カウンタユニット

図 Y 3 7 は、カウンタユニットの概略ブロック図である。カウンタユニットは、カウンタ C 1、C 2 (内部で使用) と、選択された N インデックスと、を生成する。更に、カウ

タユニットは、外部で使用するための出力フラグC 1 Z及びC 2 Zを出力する。レジスタC 1及びC 2は、D B R命令又はS C命令のターゲットであるときに更新される。オペランドのハイビット（実効コマンドのb i t₃）は、C 1とC 2の間で選択を行う。L o g i c₁及びL o g i c₂は、それぞれ、C 1及びC 2のためのW r i t e E n a b l eを判定する。

【1484】

【表195】

Logic ₁	Cycle AND (bit _{7,3} = 0x010)
Logic ₂	Cycle AND (bit _{7,3} = 0x011)

10

シングルビットフラグC 1 Z及びC 2 Zは、それらのマルチビットC 1及びC 2の対応した部分のN O R演算によって生成される。このようにして、C 1 Zは、C 1 = 0のときに1であり、C 2 Zは、C 2 = 0のときに1である。D B R命令中に、C 1又はC 2の何れかの値が1ずつデクリメントされる（ラップ付き）。デクリメンターユニットへの入力は、以下の通り、マルチプレクサM X₂によって選択される。

【1485】

【表196】

	出力	bit ₃
M X ₂	C1	0
	C2	1

20

C 1又はC 2に書き込まれた実際の値は、D B R命令又はS C命令が実行されたかどうかによって依存する。マルチプレクサM X₁は、デクリメンターからの出力（D B R命令用）と、真理表からの出力（S C命令用）との間で選択を行う。5ビット出力のうちの最下位3ビットだけがC 1に書き込まれることに注意する必要がある。マルチプレクサM X₁は、したがって、以下の選択規準をもっている。

【1486】

【表197】

	出力	bit ₆
M X ₁	真理値表からの出力	0
	デクリメンターからの出力	1

40

真理表は、S C命令を用いて、C 1及びC 2へロードされるべき値を保持する。真理表は、以下の関係を実現する簡単な組み合わせロジックである。

【1487】

【表 1 9 8】

入力値(bit ₂₋₀)	出力値
000	00010
001	00011
010	00100
011	00111
100	01010
101	01111
110	10011
111	11111

10

20

レジスタ N 1、N 2、N 3 及び N 4 は、X O R 命令によって参照されたとき、次の値 - 1 (ラップ付き) によって更新される。レジスタ N 4 は、S T

X [N 4] 命令が実行されるときにも更新される。L D 及び A D D 命令は N 4 を更新しない。その上、全部で 4 個のレジスタは S E T

N x コマンド中に更新される。L o g i c₄₋₇ は、レジスタ N 1 から N 4 の W r i t e E n a b l e を生成する。どれもが L o g i c₃ を使用し、L o g i c₃ は、コマンドが S E T

30

N x である場合に 1 を生成し、そうでなければ 0 を生成する。

【 1 4 8 8 】

【表 1 9 9】

Logic ₃	bit ₇₋₀ =01110010
Logic ₄	Cycle AND ((bit ₇₋₀ = 10101000) OR Logic ₃)
Logic ₅	Cycle AND ((bit ₇₋₀ = 10101001) OR Logic ₃)
Logic ₆	Cycle AND ((bit ₇₋₀ = 10101010) OR Logic ₃)
Logic ₇	Cycle AND ((bit ₇₋₀ = 11111011) OR (bit ₇₋₀ = 10101011) OR Logic ₃)

送出された、又はデクリメンターへの入力として使用された実際のインデックス値は、マルチプレクサ M X₄ がオペランドの下位 2 ビットを使用して簡単に選択する。

40

【 1 4 8 9 】

【表 2 0 0】

	出力	bit ₁₋₀
MX ₄	N1	00
	N2	01
	N3	10
	N4	11

10

インクリメンターは、（マルチプレクサMX₄によって選択された）入力値のうちの4ビットを取得し、1を加え、4ビット結果を生成する（モジュロ 2^4 の加法による）。最後に、マルチプレクサMX₃は4時点で、（N毎に異なり、SET N_xコマンド中にロードされる）定数値と、（XOR又はST命令中の）デクリメンターの結果と、の間で選択を行う。その値は、適当なWrite Enableフラグがセットされている場合に限り書き込まれ（Logic₄—Logic₇を参照せよ）、Logic₃はマルチプレクサのために安全に使用できる。

20

【1 4 9 0】

【表 2 0 1】

	出力	Logic ₃
MX ₃	デクリメンターからの出力	0
	定数値	1

SET N_xコマンドは、N1からN4に以下の定数をロードする。

30

【1 4 9 1】

【表 2 0 2】

インデックス	ロードされた定数	参照される初期 X [N]
N1	2	X [13]
N2	7	X [8]
N3	13	X [2]
N4	15	X [0]

40

参照される各初期X [N_n]は、最適化されたSHA-1アルゴリズムのインデックスN1からN4の初期状態と一致する。各インデックス値N_nがデクリメントするとき、実効X [N]はインクリメントする。その理由は、Xワードは最上位ワードから先にメモリに格納されているからである。3個のVALユニットは、タンパー防止及び検出回路に接続された検証ユニットであり、各VALはOKビットを含む。OKビットはリセットで1にセットされ、サイクル毎に両方のタンパー防止及び検出ラインからのChip OK値とOR演算される。OKビットはそのユニットを通過する各データビットとAND演算される。全てのVALユニットは、データビットをパリティチェックし、カウンタが改ざんされていないことを保証する。パリティチェックが失敗であるならば、消去タンパー検出ライ

50

ンがトリガーされる。VAL₁の場合、カウンタC₁からの実効出力は、チップが改ざんされているならば、常に0である。これは、攻撃者が鍵の最初から最後までインデックス付けをするループ構造を実行することを阻止する。VAL₂の場合、カウンタC₂からの実効出力は、チップが改ざんされたならば、常に0である。これは、攻撃者がループ構造を実行することを阻止する。VAL₃の場合、Nカウンタ(N₁ーN₁₄)からの実効出力は、チップが改ざんされているならば、常に0である。これは、攻撃者がXの最初から最後までインデックス付けをするループ構造を実行することを阻止する。

【1492】

次に、図203を参照すると、フラッシュメモリ記憶装置701に格納された情報705が示されている。このデータには以下のものが含まれる。

10

【1493】

工場コード

工場コードは、プリントロールが製造された工場を示す16ビットコードである。これは、プリントロールテクノロジーの所有者に属する工場、又は許可を得てプリントロールを製造する工場を識別する。この数値の目的は、品質の問題がある場合に、プリントロールの由来する工場を追跡できるようにすることである。

【1494】

バッチ番号

バッチ番号は、プリントロールの製造バッチを示す32ビット番号である。この数値の目的は、品質の問題がある場合に、プリントロールの由来するバッチを追跡することである。

20

【1495】

シリアル番号

48ビットシリアル番号は、最大で28兆個まで各プリントロールを一意に識別できるようにするため設けられる。

【1496】

製造日

16ビット製造日は、有効期限が制限されている場合に、プリントロールの経年を追跡するため組み込まれる。

【1497】

30

メディア長さ

ロール上に残っているプリントメディアの長さはこの数値によって表される。この長さは、ミリメートル、又はプリントロールを使用するプリンタ装置の最小ドットピッチのような小さい単位で表現され、周知のC、H及びPの各フォーマット、並びに、印刷されるその他のフォーマットで残り写真数の計算を可能にさせる。この小さい単位を使用することによって、プレ印刷されたメディアとの同期を保つために高解像度が使用できることを保証する。

【1498】

メディアタイプ

メディアタイプデータはプリントロールに格納されているメディアを列挙する：

40

- (1) 透明
- (2) 乳白色
- (3) 不透明着色
- (4) 3次元レンチキュラー
- (5) プレ印刷：長さ限定
- (6) プレ印刷：非長さ限定
- (7) 金属箔
- (8) ホログラフィック／光学的可変デバイス箔。

【1499】

プレ印刷メディア長さ

50

例えば、プリントロールの裏面に収容されているプレ印刷メディアの繰り返しパターンの長さがここに保存される。

【1500】

インク粘性

各インクカラーの粘性が8ビット数値として格納される。インク粘性数は、粘性を補償するためプリントヘッドアクチュエータ特性を調節する際に使用される（典型的に、粘性が高くなると、同じドロップボリュームを実現するためにはより長いアクチュエータパルスが必要になる）。

【1501】

1200dpi用の推奨ドロップボリューム

各インクカラーの推奨ドロップボリュームは8ビット数値として格納される。最も適切なドロップボリュームは、インク及び印刷メディア特性に依存する。例えば、要求されるドロップボリュームは、色素濃度又は吸収性が増加すると減少する。また、透明メディアは乳白色メディアの約2倍のドロップボリュームを必要とする。なぜならば、光は、透明メディアの場合、色素層を1回しか通過しないからである。

【1502】

プリントロールは、インクとメディアの両方を収容するので、顧客適合を実現することができる。ドロップボリュームは、推奨ドロップマッチに過ぎない。なぜならば、プリンタは1200dpiではないかもしれないし、また、プリンタは、印刷の明暗について調節されるかもしれないからである。

【1503】

インクカラー

染料色の各々のカラーが格納され、印刷前に画像に適用されるデジタル中間調を精細調整するために使用できる。

【1504】

残りメディア長さインジケータ

ロールに残っている印刷メディアの長さが数値によって表現され、カメラ装置によって更新される。長さは、C、H及びPの各フォーマット、並びに、印刷されるその他のフォーマットで残り写真数を計算できるように、小さい単位（例えば、1200dpiピクセル）で表現される。プレ印刷メディアとの同期を保つために高解像度を使用してもよい。

【1505】

著作権又はビットパターン

この512ビットパターンは、保存されたフラッシュメモリの内容を著作権保護可能にするために十分なASCII文字列を表現する。

【1506】

図204を参照すると、Artcam認証チップの記憶テーブル730が示されている。テーブルは、製造コード、バッチ番号、シリアル番号、及び日付を含み、上述のフォーマットと同じである。テーブル730は、更に、Artcam装置内のプリントエンジン上の情報731を含む。蓄積された情報は、プリンタエンジンタイプ、プリンタのDPI解像度、及びプリンタ装置によって作成されたプリント数のプリンタカウントを含む。

【1507】

更に、認証テスト鍵710が設けられ、これは、チップとチップの間でランダムに変化し、上述のアルゴリズムにおけるArtcamランダム識別コードとしても利用される。128ビットのプリントロール認証鍵713も設けられ、これはプリントロール内に保存されたキーと同等である。次に、512ビットパターンが保存され、その後に、Artcamの利用に適した120ビットの予備エリアが続く。

【1508】

上述のように、Artcamは、好ましくは、液晶ディスプレイ15を含み、液晶ディスプレイは、Artcamに格納されているプリントロールに残されたプリント数を示す。更に、Artcamは、3状態スイッチ17を含み、これにより、ユーザは、3種類の標

10

20

30

40

50

準的なフォーマット C、H 及び P（クラシック、H D T V 及びパノラマ）を切り換えることが可能である。三つの状態間の切替後、液晶ディスプレイ 15 は、選択された特定のフォーマットが使用されるならば、プリントロールに残っている画像の枚数を反映するため更新される。

【1509】

液晶ディスプレイを正しく動作させるため、A r t c a m プロセッサは、プリントロールを挿入し、認証テストに合格した後、プリントロールチップ 53 のフラッシュメモリ記憶装置から読み出し、用紙残量を判定する。次に、出力フォーマット選択スイッチ 17 の値が A r t c a m プロセッサによって判定される。プリント長さを、選択された出力フォーマットの対応した長さで除算することにより、A r t c a m プロセッサは、プリント可能数を判定し、残りプリント数で液晶ディスプレイ 15 を更新する。ユーザが出力フォーマット選択スイッチ 17 を変更すると、A r t c a m プロセッサ 31 は、そのフォーマットに応じて出力ピクチャーの数を再計算し、再度、L C D ディスプレイ 15 を更新する。

10

【1510】

プロセス情報をプリントロールテーブル 705（図 165）に保存することによって、A r t c a m 装置はプロセスの変化、及びプリントロールのプリント特性の変化を利用することができる。

【1511】

特に、プリントヘッド内の各ノズルに加えられたパルス特性は、プロセス特性の変化を考慮するために変更可能である。図 205 を参照すると、A r t c a m プロセッサは、補助メモリ R O M チップに記憶されたソフトウェアプログラムを動かすために適合し得る。ソフトウェアプログラムであるパルスプロファイルキャラクタライザ 771 は、プリントロールから多数の変数を読み出すことができる。これらの変数には、プリントロール 772 上の残りロールメディア、プリンタメディアタイプ 773、インクカラー粘性 774、インクカラードロップボリューム 775、及びインクカラー 776 が含まれる。これらの変数の各々は、パルスプロファイルキャラクタライザによって読み出され、対応した最も適しているパルスプロファイルは、従来の試行錯誤法によって決定される。パラメータは、インク出力の安定性を高めるために、各プリンタノズルで受け取られるプリンタパルスを変更する。

20

【1512】

明らかなように、認証チップは、重要かつ価値ある情報がプリンタロールと一体のプリンタチップに格納されている点で著しく進歩している。この情報は、対象となっているプリントロールのプロセス特性を、プリントロールのタイプ及びプリントロールに残っている用紙量と共に含む。

30

【1513】

更に、プリントロールインタフェースチップは、貴重な認証情報を提供し、耐タンパー性の形で構築することが可能である。また、このチップを利用する耐タンパー性のある方法が提供される。プリントロールチップの利用は、A r t c a m 装置の迅速な出力フォームのために便利かつ効率的なユーザインタフェースを提供することができ、多数の写真フォーマットを出力でき、同時に、印刷装置に残っている写真の数のインジケータを提供することができる。

40

【1514】

プリントヘッドユニット

図 206 には、図 162 のプリントヘッドユニット 615 の分解斜視部分断面図が示されている。

【1515】

プリントヘッドユニット 615 は、像を形成するように要求に応じてインクドロップをプリントメディア 611 へ噴射するプリントヘッド 44 が中心に置かれる。プリントメディア 611 は、第 1 セット 618 及び 616 と、第 2 セット 617 及び 619 からなる 2 組のローラーの間に挟まれている。

50

【1516】

プリントヘッド44は電源、グラウンド、及び信号線810の制御下で動作し、電源、グラウンド、及び信号線810は、プリントヘッド44用の電源及び制御を提供し、テープ自動ボンディング(TAB)によってプリントヘッド44の表面に接合されている。

【1517】

重要なことは、適当に分離されたシリコンウェハー装置から構築することができるプリントヘッド44は、一連の異方性エッチ812に依拠し、ウェハーを介して、ほぼ垂直な側壁を有する。ウェハーを貫通するエッチ812は、後で取り出すためウェハーの裏側からインクをプリントヘッド表面に直接供給することができる。

【1518】

インクは、インクヘッド供給ユニット814を用いてインクジェットプリントヘッド44の裏側に供給される。インクジェットプリントヘッド44は、その表面に沿って、別々の色のインクを供給するため3つの別個の行をもつ。インクヘッド供給ユニット814は、インクチャンネルをシールする蓋815を含む。

【1519】

図207から図210には、インクヘッド供給ユニット814の多数の斜視図が示されている。図207から図210の各々は、長さに制限のないインクヘッド供給ユニットの一部だけを示し、図示された部分は典型的な細部を示す。図207には、底側斜視図が示され、図148は上側斜視図を示し、図209は拡大斜視部分断面図を示し、図210はインクチャンネルの詳細を示す上側斜視図であり、図211は上側斜視図であり、図212も同様である。

【1520】

インクヘッド供給ユニット814は、例えば、マイクロマシーン加工されたシリコンからではなく、射出成形プラスチックから形成する方がコスト的にかなり有利である。プラスチックインクチャンネルの製造コストは非常に低く、製造は十分に容易である。添付図面に示された設計は、所定の長さの1600 dpiの3色モノリシックプリントヘッドを想定している。与えられた流速計算は、100 mmフォトプリンタに対するものである。

【1521】

インクヘッド供給ユニット814は、必要な精細部の全てを含んでいる。蓋815(図206)は、インクヘッド供給ユニット814に永久的に接着されるか、又は超音波溶接され、インクチャンネルのシールを提供する。

【1522】

図209を参照すると、シアン、マゼンタ、イエローのインクが、インクインレット820-822に流入し、マゼンタインクはスルーホール824, 825の中を流れ、マゼンタメインチャンネル826, 827(図141)に沿って流れる。シアンインクは、シアンメインチャンネル830に沿って流れ、イエローインクはイエローメインチャンネル831に沿って流れる。図209に最もよく示されているように、シアンインクメインチャンネル内のシアンインクは、シアンサブチャンネル833に流れ込む。イエローサブチャンネル834は、同様に、イエローメインチャンネル831からイエローインクを受け取る。

【1523】

図210の最もよく示されているように、マゼンタインクもマゼンタメインチャンネル826, 827からマゼンタスルーホール836, 837へ流れる。図209に戻ると、マゼンタインクはスルーホール836, 837から流出する。マゼンタインクは、マゼンタサブチャンネル、例えば、838に沿って流れ、次に、第2のマゼンタサブチャンネル、例えば、839に沿って流れ、その後、マゼンタトラフ840に流入する。マゼンタインクは、次に、マゼンタバイア、例えば、842を流れ、マゼンタバイアは、対応したインクジェットヘッドスルーホール(例えば、図166の812)と整列させられ、それらは、次に、インクをプリントアウト用のインクジェットノズルへ供給する。

【1524】

同様に、シアンサブチャンネル 8 3 3 内のシアンインクはシアンピットエリア 8 4 9 に流入し、それは、インクを二つのシアンバイア 8 4 3, 8 4 4 に供給する。同様に、イエローサブチャンネル 8 3 4 は、イエローピットエリア 4 6 へ供給し、イエローピットエリアはイエローバイア 8 4 7, 8 4 8 へ供給する。

【1 5 2 5】

図 2 1 0 に示されるように、プリントヘッドは、プリントヘッドスロット 8 5 0 に収容できるように設計され、種々のバイア、例えば、8 5 1 は、プリントヘッドウェハー内で対応したスルーホール、例えば、8 5 1 と揃えられる。

【1 5 2 6】

図 2 0 6 に戻ると、適切なインクがプリントヘッドチップ 4 4 の全体へ流れ、同時に、射出成形プロセスの制約を満たすように注意すべきである。プリントヘッドチップの裏側にあるインクスルーウェハーホール 8 1 2 のサイズは、約 $100\mu\text{m} \times 50\mu\text{m}$ であり、異なる色のインクを運ぶスルーホールの隙間は約 $170\mu\text{m}$ である。このサイズの形状は容易にプラスチック成形でき（コンパクトディスクはミクロンサイズの形状である）、理想的には壁の高さは、適当なスティフネスを維持するため、壁の厚さの数倍を超えてはならない。好適な実施形態は、少しずつ小さくなるインクチャンネルの階層構造を使用することによってこの問題を解決している。

【1 5 2 7】

図 2 1 1 には、プリントヘッド 4 4 の表面の小部分 8 7 0 が示されている。表面は、シアン系列 8 7 1、マゼンタ系列 8 7 2、及びイエロー系列 8 7 3 からなる 3 系列のノズルに分割される。ノズルの各系列は、更に、2 列、例えば、8 7 5、8 7 6 に分割され、プリントヘッド 4 4 は、電源及び制御信号を接合する接合パッド 8 7 8 の系列を含む。

【1 5 2 8】

プリントヘッドは、好ましくは、Artcam 装置を含む用途のため創作された非常に多数のインクジェットの異なる形態に応じて構築される。これらのインクジェット装置は以下で詳述される。

【1 5 2 9】

プリントヘッドノズルは、図 2 0 6 の異方性エッチホール 8 1 2 と同等のインク供給チャンネル 8 8 0 を含む。インクは、ウェハーの裏側から、供給チャンネル 8 8 1 を通り、次に、フィルタグリル 8 8 2 を通り、インクノズルチャンバー、例えば、8 8 3 へ流れる。ノズルチャンバー 8 8 3 及びプリントヘッド 4 4（図 1）の動作は、既に説明した通りであり、上述の特許明細書に記載されている。

【1 5 3 0】

インクチャンネル流体フロー解析

次に、インクフローの解析を説明すると、メインインクチャンネル 8 2 6、8 2 7、8 3 0、8 3 1（図 2 0 7、図 1 4 1）は、約 $1\text{mm} \times 1\text{mm}$ であり、1 色の全てのノズルに供給する。サブチャンネル 8 3 3、8 3 4、8 3 8、8 3 9（図 2 0 9）は約 $200\mu\text{m} \times 100\mu\text{m}$ であり、一つが約 25 のインクジェットノズルへ供給する。プリントヘッドスルーホール 8 4 3、8 4 4、8 4 7、8 4 8 及びウェハースルーホール、例えば、8 8 1（図 2 2 1）は、 $100\mu\text{m} \times 50\mu\text{m}$ であり、プリントヘッドスルーホールのそれぞれの側で 3 個のノズルに供給する。各ノズルフィルタ 8 8 2 は 8 スリットを有し、各スリットは $20\mu\text{m} \times 2\mu\text{m}$ の面積を有し、一つのノズルに供給する。

【1 5 3 1】

上記のように構成されたインクジェットプリンタの圧力必要条件の解析を行った。解析は、写真印刷用の 1600dpi の 3 色プロセスプリントヘッドに対するものである。プリント幅は 100mm であり、1 色毎に 6250 のノズルがあり、全部で 18750 のノズルがある。

【1 5 3 2】

フルブラック印刷の場合に種々のチャンネルで必要な最大インク流速が重要である。これは、インクチャンネルに沿った圧力降下を決定し、したがって、プリントヘッドが表面張

10

20

30

40

50

力だけで充満した状態を保つかどうか、保てない場合に、プリントヘッドを充満した状態に保つために必要なインク圧を決定する。

【1533】

圧力降下を計算するため、1600 dpi 動作に対して2.5 pI のドロップボリュームを利用した。ノズルはもっと高速に動作する能力を備えているが、選択されたドロップ繰返しレートは5 kHzであり、これは、150 mm長さの写真を2秒以内に印刷するために適当である。このようにして、極端なケースでは、プリントヘッドは、18750 ノズルを有し、すべてが毎秒最大5000 ドロップで印刷をする。このインクフローは、インクチャンネルの階層構造全体に広がる。各インクチャンネルは、全てのノズルが印刷をしているとき、事実上、一定数のノズルに供給する。

10

【1534】

圧力降下 Δp は、Darcy-Weisbach 式に従って計算した：

$$\Delta p = \rho U^2 f L / 2 D$$

ここで、 ρ はインクの密度であり、 U は平均流速であり、 L は長さであり、 D は水力直径であり、 f は無次元摩擦係数であり、以下の式で計算した：

$$f = \kappa / Re$$

ここで、 Re はレイノルズ数であり、 κ は、チャンネルの断面積に依存する無次元摩擦係数であり、以下の式で計算した：

$$Re = U D / \nu$$

ここで、 ν はインクの運動学的粘性である。

20

【1535】

矩形断面の場合、 κ は、

$$\kappa = 64 / (2/3 + (11b/24a) (11b/24a) (2 - b/a))$$

によって近似できる。

【1536】

式中、 a は矩形断面の長辺であり、 b は短辺である。矩形断面に対する水力直径 D は、

$$D = 2ab / (a + b)$$

によって与えられる。

【1537】

インクは、チャンネルの長手方向の250点でメインインクチャンネルから抜かれた。インク粘性はチャンネルの先端から線形に降下し、チャンネルの終端で0まで降下するので、平均流速 U は最大流速の半分である。したがって、メインインクチャンネルに沿った圧力降下は、最大流速を用いて計算した圧力降下の半分である。

30

【1538】

これらの式を利用して、圧力降下を計算することにより、次の表が得られた。

【1539】

インクチャンネル寸法と圧力降下の表

【1540】

【表 2 0 3】

	アイテム数	長さ	幅	深さ	供給ノズル	5KHzでの最大インク流(U)	圧力低下 Δp
中央モールディング	1	106mm	6.4mm	1.4mm	18,750	0.23ml/s	適用せず
シアンメインチャンネル(830)	1	100mm	1mm	1mm	6,250	0.16 $\mu\text{l}/\mu\text{s}$	111Pa
マゼンタメインチャンネル(826)	2	100mm	700 μm	700 μm	3,125	0.16 $\mu\text{l}/\mu\text{s}$	231Pa
イエローメインチャンネル(831)	1	100mm	1mm	1mm	6,250	0.16 $\mu\text{l}/\mu\text{s}$	111Pa
シアンサブチャンネル(833)	250	1.5mm	200 μm	100 μm	25	0.16 $\mu\text{l}/\mu\text{s}$	41.7Pa
マゼンタサブチャンネル(834)(a)	500	200 μm	50 μm	100 μm	12.5	0.031 $\mu\text{l}/\mu\text{s}$	44.5Pa
マゼンタサブチャンネル(838)(b)	500	400 μm	100 μm	200 μm	12.5	0.031 $\mu\text{l}/\mu\text{s}$	5.6Pa
イエローサブチャンネル(834)	250	1.5mm	200 μm	100 μm	25	0.016 $\mu\text{l}/\mu\text{s}$	41.7Pa
シアンビット(842)	250	200 μm	100 μm	300 μm	25	0.010 $\mu\text{l}/\mu\text{s}$	3.2Pa
マゼンタスルー(840)	500	200 μm	50 μm	200 μm	12.5	0.016 $\mu\text{l}/\mu\text{s}$	18.0Pa
イエロービット(846)	250	200 μm	100 μm	300 μm	25	0.010 $\mu\text{l}/\mu\text{s}$	3.2Pa
シアンピア(843)	500	100 μm	50 μm	100 μm	12.5	0.031 $\mu\text{l}/\mu\text{s}$	22.3Pa
マゼンタピア(842)	500	100 μm	50 μm	100 μm	12.5	0.031 $\mu\text{l}/\mu\text{s}$	22.3Pa
イエローピア	500	100 μm	50 μm	100 μm	12.5	0.031 $\mu\text{l}/\mu\text{s}$	22.3Pa
マゼンタスルーホール(837)	500	200 μm	500 μm	100 μm	12.5	0.003 $\mu\text{l}/\mu\text{s}$	0.87Pa
チップスロット	1	100mm	730 μm	625	18,750	適用せず	適用せず
プリントヘッドスルーホール(881) (チップ基板内)	1500	600 μm	100 μm	50 μm	12.5	0.052 $\mu\text{l}/\mu\text{s}$	133Pa
プリントヘッドチャンネルセグメント(オンチップフロント)	1色 当たり 1000	50 μm	60 μm	20 μm	3,125	0.049 $\mu\text{l}/\mu\text{s}$	62.8Pa
フィルタスリット (ノズル室への入口) (882)	ノズル 毎に 8	2 μm	2 μm	20 μm	0.125	0.039 $\mu\text{l}/\mu\text{s}$	251Pa
ノズル室 (オンチップフロント) (883)	ノズル 毎に 1	70 μm	30 μm	20 μm	1	0.021 $\mu\text{l}/\mu\text{s}$	75.4Pa

インクインレットからノズルまでの総圧力降下は、したがって、シアン及びイエローに関して約 7 0 1 P a であり、マゼンタに関して 8 4 5 P a である。これは、大気圧の 1 % 未満である。勿論、印刷された画像がフルブラック未満の場合、インクフロー（したがって、圧力降下）はこれらの値よりも低下するであろう。

【1 5 4 1】

インクヘッド供給ユニット用の金型製作

インクヘッド供給ユニット 1 4（図 1）は 5 0 μ の外形と 1 0 6 m m の長さがある。射出成形ツールを従来の方法で加工することは実施不可能である。しかし、全体的な形状が複雑であるとしても、複雑な曲線は不要である。射出成形ツールは、メインインクチャンネル及びその他のミリメートルスケールの形状については従来のミリングを使用して製作可能であり、精細な形状はリソグラフィ的に製作されたインセットを用いる。L I G A プロセスをインセットのために使用できる。

【1 5 4 2】

単一の射出成形ツールは、簡単に 5 0 以上のキャビティを含む。ツールの複雑さの殆どは

インセットにある。

【1543】

図206を参照すると、印刷システムはインク供給ユニット814及び蓋815を一体的に成形し、上述のようにそれらを一体的にシーリングする。次に、プリントヘッド44が対応したスロット850に設置される。粘着性シーリングストリップ852、853がマゼンタメインチャンネル上に置かれ、それらが適切にシールされることを保証する。テープ自動ボンディング(TAB)ストリップ810がインクジェットプリントヘッド44に接続され、タブボンディングワイヤはキャビティ855内を通る。図206、図207及び図212からよく分かるように、アパーチャスロット855から862は、ローラーの挿入の際のスナップのために設けられる。スロットは、ある程度の遊びをもってローラー

10

【1544】

図213から図217には、最終的に組み立てられたArtcam装置の内部の様々な斜視図が示され、装置には適当な番号が付されている：

- ・図213は、Artcamカメラの内部の上面側の斜視図であり、平らにされた部品が示されている；
- ・図214は、Artcamカメラの内部の底面側の斜視図であり、平らにされた部品が示されている；
- ・図215は、Artcamカメラの内部の第1の上面側の斜視図であり、Artcamに収容された部品が示されている；
- ・図216は、Artcamカメラの内部の第2の上面側の斜視図であり、Artcamに収容された部品が示されている；
- ・図217は、Artcamカメラの内部の第2の上面側の斜視図であり、Artcamに収容された部品が示されている。

20

【1545】

ポストカードプリントロール

図218を参照すると、好適な実施形態の一形態として、出力プリンタ用紙11は、画像が印刷されない面に、多数のプレ印刷された「ポストカード」型裏書き部分885を含む。ポストカード型セクション885は、プリペイド式郵便料金「スタンプ」886を含み、この「スタンプ」886は、関連した郵便当局からの印刷された認証により構成され、その当局の管轄区域内で、そのプリントロールが販売若しくは利用される。関連した管轄区域の郵便当局との合意によって、プリントロールは様々な郵便料金を利用できる。これは、特に、外国からの旅行者が国内の管轄内にいて、自国へ多数のポストカードを送る場合に便利である。更に、住所書式部分887が設けられ、通常のポストカードの形式で宛先発送の詳細を記入できる。最後に、メッセージ領域887が設けられ、私信を記入することができる。

30

【1546】

図218及び図219を参照して、カメラ装置の動作を説明すると、一連の画像890から892がプリントロールの第1面に印刷されたときの、対応した裏書き面が図218に示されている。したがって、各画像、例えば、891がカメラによって印刷されると、画像の裏面は、既にポストカード885になっているので、直ちに、管轄区域内の最も近い郵便ポストへ投函することができる。このように、パーソナル化されたポストカードを作成することができる。

40

【1547】

図219及び図220に示されるようなポストカードシステムを利用するとき、予め定められた画像サイズだけが可能であることは明白である。なぜならば、裏面のポストカード部分885と、表面の画像891との対応関係を維持しなければならないからである。これは、各ポストカードの裏書き書式シート885の長さの詳細を記憶するためプリントロール内に収容された認証チップのメモリ部分を利用することによって実現される。これは、各ポストカードのサイズを同じサイズにすること、又はそれぞれのサイズをプリントロ

50

ールのオンボードプリントチップメモリに格納することによっても実現される。

【1548】

Artcamカメラ制御システムは、予め書式化されたポストカードを利用するとき、プリントロールは、各画像がポストカードの境界に収まるように画像を印刷するためだけに使用されることを保証することができる。勿論、多少の位置合わせ誤差を考慮できるように境界領域を各写真のエッジに設けることによってある程度の「遊び」が与えられる。

【1549】

図220から明らかであるように、ポストカードロールは、それを利用可能な特定の管轄区域内に旅行するとき、カメラユーザが事前に購入できる。ポストカードロールは、その外面に、購入国、各ポストカードの郵便料金、各ポストカードのフォーマット（例えば、C、H、P、又はこれらの画像モードの組み合わせ）、使用するのに適した国名、及びその日以降に郵便料金が不足になるかもしれない郵便料金期限を含む情報が印刷されている。

10

【1550】

したがって、カメラ装置のユーザは、関連したシーンに向けられた自分のハンドヘルドカメラを利用して、写真を撮影して、その画像が一方の面に掲載され、もう一方の面にはプリペイド式ポストカードの詳細が記載されている郵便で発送するためのポストカードを製作することが可能である。次に、このポストカードに宛先を指定し、短いメッセージを記入し、その後、直ぐに郵便で発送することができる。

【1551】

Artcam装置のソフトウェア動作に関して、種々のソフトウェア設計が可能であるが、ある設計では、各Artcam装置は、1組の緩く結合した機能モジュールにより構成され、これらの機能モジュールは、一つの埋め込み型アプリケーションによって連係した形で利用できるようにされ、装置の中心目的のために機能する。機能モジュールは、様々なクラスのArtcam装置において、異なる組み合わせで再使用され、アプリケーションはArtcamのクラスに特有である。

20

【1552】

殆どの機能モジュールは、ソフトウェアとハードウェアの両方のコンポーネントを含む。ソフトウェアは、ハードウェア抽象化レイヤによって、ハードウェアの細部からシールドされ、一方、モジュールのユーザは、抽象化ソフトウェアインタフェースによってソフトウェアインプリメンテーションからシールドされている。システムは、全体として、ユーザ始動、及びハードウェア始動のイベントによって駆動されるので、殆どのモジュールは、一つ以上の同期的なイベント駆動型プロセスを動かすことができる。

30

【1553】

一般的なArtcam装置を構成する殆どの重要なモジュールが図221に示されている。同図及びその後の図には、ソフトウェアコンポーネントが左側に示めされ、垂直点線901によって右側のハードウェアコンポーネントから分離されている。これらのモジュールのソフトウェアの面を次に説明する。

【1554】

ソフトウェアモジュール—Artcamアプリケーション902

40

ArtcamアプリケーションはArtcam装置の上位レベル機能を実現する。これは、通常、像の撮影、像に対する芸術的効果の適用、及び像の印刷を含む。カメラ指向のArtcam装置の場合、画像は、カメラマネージャ903によって捕捉される。プリンタ指向のArtcam装置の場合、画像は、ネットワークマネージャ904によって捕捉され、おそらく、この画像は他の装置によって送出された結果物である。

【1555】

芸術的効果は、ファイルマネージャ905によって管理される統合ファイルシステム内で行われる。芸術的効果は、スクリプトファイルとリソースのセット途により構成される。スクリプトは、画像処理マネージャ906によって翻訳され、画像に適用される。スクリプトは、通常、Artcardとして知られているArtCardsで出荷される。デフ

50

ォルトとして、システムは、現在搭載されている A r t c a r d に格納されているスクリプトを使用する。

【1556】

画像はプリントマネージャ908によって印刷される。

【1557】

A r t c a m 装置がスタートアップしたとき、ブートストラッププロセスが、アプリケーションを開始する前に種々のマネージャプロセスを始動する。これにより、アプリケーションは、スタートしたときに、直ちに様々なマネージャからのサービスを要求できる。

【1558】

初期化後、アプリケーション902は、それ自体を以下に列挙されたイベントのハンドラーとして登録する。アプリケーションはイベントを受け取ったとき、表に記載された動作を実行する。

【1559】

【表204】

ユーザインタフェースイベント	動作
焦点固定	カメラマネージャによって自動ブレ撮影セットアップを実行。これには、自動焦点機能、自動露出調整、及び、フラッシュの充電が含まれる。これは、通常、ユーザが撮影ボタンを半押しすることによって起動される。
撮 影	カメラマネージャによって撮像。
セルフタイマー	カメラマネージャによってセルフタイマーモードで撮像。
フラッシュモード	次のフラッシュモードを使用するためカメラマネージャを更新。新しいフラッシュモードを表示するため状態ディスプレイを更新。
プリント	プリントマネージャによって現在画像を印刷。現在スクリプトが存在する場合、画像処理マネージャによって画像に芸術的效果を適用。状態ディスプレイ上の残りプリントカウントを更新(以下のプリントロール装填を参照せよ。)
ホールド	現在スクリプトが存在する場合、画像処理マネージャによって画像に芸術的效果を適用するが、画像を印刷しない。
ArtCard 取り出し	ファイルマネージャによって現在装填されている ArtCard を取り出し。
プリントロール装填	印刷マネージャの残りメディア長さとかメラマネージャのアスペクト比とに基づいて残りプリント数を計算。状態ディスプレイ上の残りプリントカウントを更新。
プリントロール取り外し	プリントロールが装着されていないことを報せるため状態ディスプレイを更新。

カメラがディスプレイを含むとき、アプリケーションはユーザインタフェースマネージャ910によってグラフィカルユーザインタフェースを構築し、ユーザは現在日時、及びその他の編集可能なカメラパラメータを編集できるようになる。アプリケーションは、全ての常駐パラメータをフラッシュメモリに保存する。

【1560】

リアルタイムマイクロカーネル911

リアルタイムマイクロカーネルは、割込とプロセス優先順位に基づいて、予めプロセスをスケジューリングする。それは、プロセススケジューリングに非常に密接している統合プロセス間通信及びタイマーサービスを提供する。全ての他のオペレーティングシステム機能はマイクロカーネルの外部で行われる。

【1561】

カメラマネージャ903

カメラマネージャは、画像捕捉(撮像)サービスを提供する。それは、A r t c a m に組み込まれたカメラハードウェアを制御する。それは、抽象的カメラ制御インタフェースを提供し、カメラパラメータの問い合わせ及び設定、画像捕捉を可能にさせる。この抽象的

インタフェースは、アプリケーションを他のカメラインプリメンテーションの細部から切り離す。カメラマネージャは以下の入力／出力パラメータ及びコマンドを利用する。

【 1 5 6 2 】

【表 2 0 5】

出力パラメータ	ドメイン
焦点レンジ	実数、実数
ズームレンジ	実数、実数
絞りレンジ	実数、実数
シャッター速度レンジ	実数、実数

10

【 1 5 6 3 】

【表 2 0 6】

入力パラメータ	ドメイン
焦点	実数
ズーム	実数
絞り	実数
シャッター速度	実数
アスペクト比	標準、HDTV、パノラマ
焦点制御モード	多点オート、単点オート、マニュアル
露出制御モード	オート、絞り優先、シャッター優先、マニュアル
フラッシュモード	オート、レッドアイ除去付きオート、充電、オフ
ビューシーンモード	オン、オフ

20

【 1 5 6 4 】

【表 2 0 7】

コマンド	返値ドメイン
焦点固定	無し
セルフタイマー撮影	未加工画像
撮像	未加工画像

30

カメラマネージャは、非同期イベント駆動型プロセスとして動く。それは、連結された状態機械の組を含み、各状態機械は一つの非同期オペレーションに対応する。それらは、オートフォーカス、フラッシュ充電、セルフタイマのカウントダウン、及び像の撮影を含む。初期化時、カメラマネージャはカメラハードウェアを既知状態にセットする。これは、通常の焦点距離の設定、及びズームの戻しを含む。カメラマネージャのソフトウェア構造は、図 2 2 2 に示されている。ソフトウェアコンポーネントは以下のサブセクションに記載されている。

40

【 1 5 6 5 】

フォーカス固定 9 1 3

ロックフォーカスは、フォーカス制御モード、露出制御モード、及びフラッシュモードに応じて、現在シーンに対して自動的にフォーカスと露出を調整し、必要に応じてフラッシュを動作可能状態にする。ロックフォーカスは、通常、ユーザが撮影ボタンを半押しすることによって始動される。これは、正常な像撮影シーケンスの一部であるが、ユーザが撮影ボタンを半押しのままにした場合、時間的に、実際の像の撮影からは分離される。これは、ユーザがスポットフォーカシング及びスポット測距をできるようにする。

【 1 5 6 6 】

撮影 9 1 4

50

撮影（像捕捉）は、現在シーンの像を撮影する。フラッシュモードがレッドアイ除去を含む場合、レッドアイランプを点灯し、シャッターを制御し、動作可能状態であれば、フラッシュをトリガーし、イメージセンサによって像を検知する。それは、カメラの向き、即ち、撮影像の向きを判定し、これにより、後の画像処理で、像は適切に向きを決められる。また、それは、後の画像処理中にブラーを取り除くため、撮像中にカメラの動きの有無を検出する。

【1567】

セルフタイム撮影 915

セルフタイム撮影は、20秒タイマーのカウントダウン後に、現在シーンの像を撮影する。これは、カウントダウン中に、セルフタイマーLEDを用いて、ユーザにフィードバックを与える。最初の15秒間に、それは、LEDを点灯できる。最後の5秒間にLEDは点滅する。

10

【1568】

ビューシーン 917

ビューシーンは、周期的に、イメージセンサを通して現在シーンを検知し、そのシーンをカラーLCDに表示させ、ユーザにLCDベースのビューファインダーを提供する。

【1569】

オートフォーカス 918

オートフォーカスは、像の選択された領域が焦点が合っていると言えるまで十分に鮮明になるまで、焦点距離を変化させる。それは、イメージセンサの指定された領域から抽出された画像鮮明さメトリックが固定の閾値を超えた場合に領域の焦点が合うということを前提とする。それは、鮮明さの微分に関する勾配降下法を実行し、必要に応じて方向及びステップサイズを変えることによって、最適焦点距離を見つける。焦点制御モードがマルチポイントオートであるならば、視野全体に水平に配置された3つの領域が使用される。焦点制御モードがシングルポイントオートであるならば、視野の中心にある一つの領域が使用される。オートフォーカスは、フォーカスコントローラによって指定された有効焦点距離の範囲内で機能する。固定焦点装置の場合、これは、事実上、無効にされる。

20

【1570】

オートフラッシュ 919

オートフラッシュは、シーンの照度がフラッシュを必要とする程度に薄暗いかどうかを判定する。これは、シーンの照度が所定の閾値以下であるならば、照度が十分に薄暗いとみなす。シーンの照度は、照度センサから獲得され、照度センサはイメージセンサの中央領域から照度メトリックを抽出する。フラッシュが必要であれば、フラッシュを充電する。

30

【1571】

自動露出 920

シーン照度、絞り、および、シャッター速度は、撮影像の露出を決定する。望ましい露出は固定値である。露出制御モードがオートであるならば、自動露出は、所定のシーン照度に対して望ましい露出を生じる絞りとシャッター速度の組み合わせを決定する。露出制御モードが絞り優先であるならば、自動露出は、所定のシーン照度及び現在の絞りに対して望ましい露出を生じるシャッター速度を決める。露出制御モードがシャッター優先であるならば、自動露出は、所定のシーン照度及び現在のシャッター速度に対して望ましい露出を生じる絞りを決める。シーン照度は照度センサから取得され、照度センサはイメージセンサの中央領域から照度メトリックを抽出する。

40

【1572】

自動露出は、絞りコントローラ及びシャッター速度コントローラによって指定された有効絞り及びシャッター速度の範囲内で機能する。シャッター速度コントローラ及びシャッターコントローラは、殆どのArtcam装置では機械的シャッターの無いことを隠す。

【1573】

フラッシュが、マニュアル又はオートで有効にされている場合、実効的なシャッター速度はフラッシュの間隔であり、これは、典型的に、1/1000秒から1/10000秒で

50

ある。

【1574】

画像処理マネージャ906 (図221)

画像処理マネージャは、画像処理と芸術的効果サービスを提供する。これは、高速画像処理を実行するためArtcamに組み込まれたVLWベクトルプロセッサを利用する。画像処理マネージャは、Vark画像処理言語で記述されたスクリプト用のインタープリタを具備する。したがって、芸術的効果は、Varkスクリプトファイルと、フォント、クリップ画像のような関連したリソースと、を含む。画像処理マネージャのソフトウェア構造は図223により詳細に示され、以下のモジュールを含む。

【1575】

画像の変換及び強調921

画像処理マネージャは、装置に依存しないCIE L A B色空間において、Artcamプリンタハードウェアの生成能力に適した解像度で、画像処理を実行する。撮影された像は、最初に、ノイズをフィルタ処理して強調される。それは、オブションとして、動きによって生じたブラー効果を取り除くために行われる。画像は、次に、装置に依存したRGB色空間から、CIE

L A B色空間へ変換される。像は、撮影時のカメラ回転の影響を相殺するため回転させられ、作業用画像解像度へスケール変換される。画像は、そのダイナミックレンジを利用可能なダイナミックレンジへスケールリングすることによって強調される。

【1576】

顔検出923

顔は、色相及び局部的特徴解析に基づいて撮影された像内で検出される。検出された顔領域のリストは、ワーピング及び吹き出しの配置のような顔専用効果を適用するためVarkスクリプトによって使用される。

【1577】

Vark画像処理言語インタープリタ924

Varkは、豊富な画像処理のセットで拡張された汎用プログラミング言語を含む。それは、プリミティブデータタイプの範囲（整数、実数、ブール、文字）、より複雑なタイプを構築するための集合データタイプの範囲（配列、文字列、レコード）、算術演算及び論理演算の豊富な組、条件付き及び反復制御フロー（if-then-else、while-do）、及び再帰的関数及び手続を提供する。また、それは、画像処理データタイプの範囲（画像、クリップ画像、マット、カラー、カラーlookupアップテーブル、パレット、ディザ行列、コンボリューションカーネル、等）、グラフィックデータタイプ（フォント、テキスト、パス）、画像処理関数の組（カラー変換、コンポジット処理、フィルタ処理、空間変換及びワーピング、イルミネーション、テキスト設定及びレンダリング）、及び上位レベルの芸術的関数の組（タイリング、ペインティング、及びストローキング）を提供する。

【1578】

Varkプログラムは二つの意味でポータブルである。それは、インタープリットされるので、CPU及びホストの画像処理エンジンに依存しない。装置に依存しないモデル空間及び装置に依存しない色空間を使用するので、それは、入力色特性、ホスト入力装置の解像度、出力色特性、及びホスト出力装置の解像度に依存しない。

【1579】

Varkインタープリタ924は、Varkスクリプトを構成するソースステートメントを解析し、そのスクリプトの意味を表現するパースツリーを生成する。パースツリーのノードは、プログラムのステートメント、エクスプレッション、サブエクスプレッション、変数、及び定数に対応する。ルートノードはメインプロシジャーステートメントリストに対応する。このインタープリタは、パースツリーのルートステートメントを実行することにより、プログラムを実行する。パースツリーの各ノードはその子にそれ自体を適切に評価又は実行するように要求する。例えば、ifステートメントノードは、条件表現ノード

10

20

30

40

50

と、`then`ステートメントノードと、`else`ステートメントノードの三つの子を持つ。`if`ステートメントは、条件表現ノードに、それ自体を評価し、返されたブール値に基づいて、`then`ステートメント又は`else`ステートメントにそれ自体を実行させる。それは、実際の条件表現又は実際のステートメントは知らない。

【1580】

殆どのデータタイプの演算はパースツリーの実行中に行われるが、画像データタイプの演算は、パースツリーの実行後まで、延期される。これにより、画像演算が最適化され、最終的な画像に寄与する中間画素だけが計算される。これは、最終画像を、空間細分によって多数のパスで計算することを可能にさせ、これにより、必要なメモリ量を減少させる。

【1581】

パースツリーの実行中に、各画像関数は画像グラフを返すだけであり、この画像グラフは、ノードが画像オペレータを表し、リーフが画像を表すものであり、対応した画像オペレータをルートとし、その画像パラメータをルートの子とすることによって構築される。画像パラメータは勿論それ自体が画像グラフである。このようにして、それぞれの逐次的な画像関数は、より深い画像グラフを返す。

【1582】

パースツリーの実行後、最終画像に対応した画像グラフが得られる。この画像グラフは、次に、（任意の表現ツリーと同じように）深さ優先方式で実行され、以下の2点で最適化される。（1）最終画像に寄与する画素だけが所定のノードで計算される。（2）ノードの子は、メモリ必要量を最小限に抑える順番で実行される。画像グラフ中の画像オペレータは、最終画像を生成するために最適化された順序で実行される。計算集約的な画像オペレータは、`Artcam`装置に組み込まれた`VLIW`プロセッサを使用して加速される。画像グラフを実行するために必要なメモリ量が利用可能なメモリを超えたとき、最終画像領域は、必要なメモリが利用可能なメモリを超えなくなるまで細分される。

【1583】

巧く構築された`Vark`プログラムの場合、最初の最適化は特に大きな効果があるとは思えない。しかし、最終画像領域が細分された場合、最適化はかなりの効果を生ずる可能性がある。細分化がメモリ必要量を減少させる効果的な技術として使用することができるが、まさにこの最適化である。画像オペレーションの実行を遅らせることの結果の一つは、プログラム制御フローを画像内容に依存させられないことである。なぜならば、画像内容はパースツリー実行中には分からないからである。實際上、これはそれほど制約にはならないが、それにもかかわらず、言語設計中には注意しなければならない。

【1584】

画像オペレーションの延期実行（又は遅延評価）の考え方については、`Guibas and Stolfi`によって記載されている（`Guibas, L. J., and J. Stolfi, 'A Language for Bitmap Manipulation'`, `ACM Transactions on Graphics, Vol. 1, No. 3, July 1982, pp. 191-214`）。彼らは、プログラムの実行中に同様に画像グラフを構築し、後続のグラフ評価中に、最終画像に寄与しない画素の計算を回避するため、結果領域を逆伝搬させる。`Shantzis`は、更に、利用可能な画素を画像グラフ評価中に前方へ伝搬させる（`Shantzis, M. A., 'A Model for Efficient and Flexible Image Computing'`, `Computer Graphics Proceedings, Annual Conference Series, 1994, pp. 147-154`）。`Vark`インタープリタは、`Cameron`によって提案された、より洗練されたマルチパス型ビットディレクショナル領

10

20

30

40

50

域伝搬スキームを使用する (Cameron, S., 'Efficient Bounds in Constructive Solid Geometry', IEEE Computer Graphics & Applications, Vol. 11, No. 3, May 1991, pp. 68-74)。メモリ使用量を最小限に押さえるため実行順序を最適化する方法は、Shantzisに由来するが、標準的なコンパイラ理論に基づいている (Aho, A. V., R. Sethi, and J. D. Ullman, 'Generating Code from DAGs', in Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986, pp. 557-567)。Varkインタープリタは、Shantzisよりも洗練されたスキームを使用するが、可変サイズ画像バッファをサポートする。メモリ使用量を減少させるため、領域伝搬と組み合わせて結果領域を細分することは、Shantzisに由来している。

10

20

30

40

【1585】
プリンタマネージャ908 (図221)
プリンタマネージャは画像プリンティングサービスを提供する。それは、Artcamに組み込まれたインクジェットプリンタハードウェアを制御する。それは、抽象的プリンタ制御インタフェースを提供し、プリンタパラメータの問い合わせと設定を可能にさせ、画像を印刷される。この抽象的インタフェースは、アプリケーションをプリンタインプリメンテーションの詳細から切り離し、以下の変数を含む。

【1586】

【表208】

出力パラメータ	ドメイン
メディア有り	ブール
固定ページサイズメディア	ブール
メディア幅	実数
残りメディア長	実数
固定ページサイズ	実数、実数

【1587】

【表209】

入力パラメータ	ドメイン
ページサイズ	実数、実数

【1588】

【表 2 1 0】

コマンド	返値ドメイン
画像印刷	無し

【 1 5 8 9】

【表 2 1 1】

出力イベント
無効メディア
メディア消耗
メディア装填
メディア取り外し

10

20

プリンタマネージャは非同期イベント駆動型プロセスとして実行される。それは、連結された状態機械の組を含み、各状態機械は一つの非同期オペレーションに対応している。これらは、画像を印刷し、プリントロールを自動マウントする。プリンタマネージャのソフトウェア構造は図 2 2 4 に示されている。ソフトウェアコマンドは後述される。

【 1 5 9 0】

プリントイメージ 9 3 0

プリントイメージは供給された画像を印刷する。これは、印刷用の画像を準備するため V L I W プロセッサ回路を使用する。V L I W プロセッサ回路は、画像色空間を装置固有 C M Y に変換し、プリントヘッドで期待されるフォーマットで中間調バイレベルデータを生成する。

30

【 1 5 9 1】

プリントの間に、プリントロールを取り外すことができるように、用紙はプリントロールのリップに引き込まれ、ノズルはインク漏れ及び乾燥を防ぐためにキャップを被せることができる。実際の印刷がスタートする前に、従って、ノズルはキャップを取り外され、きれいにされ、用紙はプリントヘッドへ進められる。印刷自体は、画像が完全に印刷されるまで、V L I W プロセッサからラインデータを転送し、ラインデータを印刷し、用紙を進める。印刷の終了後、用紙は裁断機で切断され、プリントロールに引き込まれ、ノズルにキャップが被せられる。残りメディア長さはプリントロールで更新される。

【 1 5 9 2】

プリントロール自動マウント 1 3 1

プリントロール自動マウントは、プリントロールの装着と取り外しに応答する。それは、プリントロールの装着と取り外しのイベントを生成し、このイベントは、アプリケーションによって処理され、状態表示を更新するため使用される。プリントロールは、プリントロールに組み込まれた認証チップと、A r t c a m に組み込まれた認証チップとの間でプロトコルに従って認証される。プリントロールが認証を失敗した場合、プリントロールは拒絶される。種々の情報がプリントロールから抽出される。用紙及びインク特性は、印刷プロセス中に使用される。残りメディア長さ及びメディアの固定ページサイズは、いずれにしても、プリントマネージャにより公表され、アプリケーションによって使用される。

40

【 1 5 9 3】

ユーザインタフェースマネージャ 9 1 0 (図 2 2 1)

50

ユーザインタフェースマネージャは図 2 2 5 により詳細に示され、ユーザインタフェース管理サービスを提供する。それは、状態表示及び入力ハードウェアを制御し物理的ユーザインタフェースマネージャ 9 1 1 と、カラーディスプレイ上の仮想グラフィカルユーザインタフェースを管理するグラフィカルユーザインタフェースマネージャ 9 1 2 と、を含む。ユーザインタフェースマネージャは、仮想及び物理的入力をイベントに翻訳する。各イベントはそのイベントのために登録されたプロセスのイベントキュー入れられる。

【1 5 9 4】

ファイルマネージャ 9 0 5 (図 2 2 2)

ファイルマネージャは、ファイルマネージメントサービスを提供する。それは、統合階層ファイルシステムを提供し、その中では、全てのマウントされたボリュームのファイルシステムが現れる。A r t c a m で使用される一次着脱式記憶媒体は A r t C a r d である。A r t c a r d は、2 値のドットのブロックによって高解像度で印刷され、2 値のドットのブロックは、誤差を許容するリード・ソロモン符号化 2 値データを直接表現する。ブロック構造は、(A r t c a m で初期には使用されていない) 適当なリード・ライト A r t C a r d 装置における追記及び追記一書き換えをサポートする。より高いレベルでは、A r t C a r d は、拡張型追記一書き換え可能 I S O 9 6 6 0 準拠 C D - R O M ファイルシステムを含み得る。ファイルマネージャのソフトウェア構造、及び特に、A r t C a r d 装置コントローラは、図 2 2 6 に示されている。

【1 5 9 5】

ネットワークマネージャ 9 0 4 (図 2 2 2)

ネットワークマネージャは、赤外線 (I r D A) 及び汎用シリアルバス (U S B) を含む様々なインタフェース上での家電製品ネットワークングサービスを提供する。これは、A r t c a m システムが撮影された画像を共有し、印刷のための画像を受信できる。

【1 5 9 6】

クロックマネージャ 9 0 7 (図 2 2 2)

クロックマネージャは日付及び時刻のクロックサービスを提供する。それは、A r t c a m に組み込まれたバッテリーバックアップ型リアルタイムクロックを利用し、ユーザが時刻をセットしたときに実行される自動キャリブレーションに基づいて、クロック変動を自動的に調節する程度までそれを制御する。

【1 5 9 7】

パワーマネージメント

システムが使用されていないとき、システムは、入力イベントの周期的走査が行われる休止電力状態に入る。入力イベントは、ボタンの押下、又は A r t C a r d の挿入を含む。入力イベントが検出されると直ぐに、A r t c a m 装置は、動作電力状態に戻る。その後、システムは入力イベントをいつもの通りに取り扱う。

【1 5 9 8】

システムが動作電力状態であるときでさえ、個々のモジュールに関連したハードウェアは、典型的に窮乏電力状態にある。これは、全体的な電力消費を削減し、プリンタの用紙切断機のような特に浪費のもとになるハードウェアコンポーネントが動作中に電源を独占できるようにする。カメラ指向の A r t c a m 装置は、デフォルトで、画像撮影モードである。即ち、カメラは動作状態であり、プリンタのような他のモジュールは休止状態である。つまり、カメラ以外の機能が始動されたとき、アプリケーションはカメラモジュールを明白に一時停止させなければならない。他のモジュールは、待機状態になると、自発的に一時停止する。

【1 5 9 9】

ウォッチドッグタイマー

システムは、周期的な優先度の高いウォッチドッグタイマー割込を生成する。割込ハンドラーは、システムが最後の割込以降、進んでいない、即ち、システムが故障していると判断した場合、システムをリセットする。

【1 6 0 0】

代替プリントロール

他の一実施形態では、適切に簡単に一つにはめ込まれた射出成形プラスチック部品によって大部分を構築することができる改良された形態のプリントロールが提供される。改良された形態のプリントロールは、多少構造が単純化されると共に、インク貯蔵容量が増加している。画像が印刷されるべきプリントメディアは、構造を簡単化するため、プラスチック製スリーブフォーマーの周りに巻き付けられる。インク媒体リザーバは、インクが流れ出る可能性ができるだけ少なくなる構造にされた一連の通気孔を有する。更に、ラバーシールがインク出口孔に設けられ、ラバーシールはプリントロールをカメラシステムに挿入する際に穿刺される。更に、プリントロールは、プリントメディア排出スロットを含み、排出スロットは周囲に成形された表面を有し、その表面は、プリントメディア放出スロットが、印刷又はカメラシステム内のプリントヘッドに対して正確な位置に置かれるように補助する。

10

【1601】

次に、図227から図231を参照すると、図227には、組み付けられた形式のシングルポイントロールユニット1001が示され、部分的に切断されてプリントロールの内部が表されている。図228及び図229は、それぞれ、左側及び右側の分解斜視図である。図230及び図231は図227から図229の内部コア部1007の拡大斜視図である。

【1602】

プリントロール1001は、内部インク補充部を含む内部コア部1007の周りに構築される。コア部1007の外側にはフォーマー1008が設けられ、フォーマーの周りに、用紙又はフィルム補給品1009が巻き付けられる。用紙補給品の周りには、二つのカバー部品1010、1011が設けられ、プリントロールの周りで一体としてスナップ式に留まり、図227に示されるようにカバーユニットを形成する。下側カバー部品1011はスロット1012を含み、そのスロットを通して、プリントメディア1004の出力がカメラシステムと相互連結する。

20

【1603】

二つのピンチローラー1038、1039が用紙を駆動ピンチローラー1040に圧迫するため設けられるので、それらは、一体として、ローラー1040の周りで用紙のカールを取り去る。このカールの取り除きは、長期間に亘ってプリントロールの形で蓄えられている用紙に加わる強いカールを打ち消すように作用する。ローラー1038、1039は、カバーベース部1077の端部及びローラーとスナップ式にフィットするため設けられ、ローラー1040は、駆動用の歯車付きの端部1043を含み、上側カバー部品1010とスナップ式にフィットし、用紙1004を間でしっかりと締め付ける。

30

【1604】

カバー部品1011は、端部隆起又はリップ1042を含む。端部リップ1042は用紙の出口孔を、カメラシステム内の対応した印刷熱プラテン構造体と正確にアライメントするため設けられる。このようにして、既存の用紙は、近傍のプリントヘッドに対して、正確にアライメント又は位置決めされ、用紙をプリントヘッドへ完全に案内する。

【1605】

図230及び図231を参照すると、射出成形部品から形成可能である内部コア部の分解斜視図が示され、このコア部は、内部スポンジ部分1034から1036を有する3個のコアインクシリンダが中心に置かれている。

40

【1606】

コア部の一端には、一連の通気チャンネル、例えば、1014から1016が設けられる。各通気チャンネル1014から1016は、第1の孔、例えば、1018を外部接点1019と相互連結し、外部接点は雰囲気と相互連結される。通気チャンネル、例えば、1014が後に続く通路は、好ましくは、曲がりくねった性質があり、あちこちに曲がりくねっている。通気チャンネルは、シーリングテープ1020の一部分によってシールされ、シーリングテープはコア部の端に被せられる。シーリングテープ1020の表面は、好

50

ましくは、疎水性を高めるため、疎水性的に処理されているので、流体部分が通気チャンネルに進入することを防ぐ。

【1607】

コア部1007の第2の端には、3個の厚くなった部分1024、1025及び1026を含むラバーシーリングキャップ1023が設けられ、厚くなった各部分は一連の疎らな孔が存在する。例えば、部分1024は、疎らな孔1029、1030及び1031を含む。疎らな孔は、別個の厚くなった部分の各々からの1個の孔は1本の直線に並ぶように配置される。例えば、疎らな孔1031、1032及び1033（図230）は、全てが1本の線に並び、各孔は別個の厚くなった部分に由来する。厚くなった部分の各々は、対応したインク供給リザーバと対になり、3個の孔が穿刺されたとき、対応したリザーバと流体連通する。

10

【1608】

端部キャップユニット1044は、コア部1007に取り付けられる。端部キャップ1044は、認証チップ1033を挿入するためのアパーチャ1046を含み、その上、3個の突起を含む延長されたアダプタ（図示せず）を含み、3個の突起は、対応した孔（例えば、1048）を貫通し、シール1023の厚くなった部分（例えば、1033）を穿刺し、対応したインクチャンバー（例えば、1035）と相互連結する。

【1609】

更に、端部1044には、認証チップ1033が挿入され、認証チップは、プリントロールのカメラシステムへのアクセスを認証するため設けられる。したがって、このコア部は、3個の別個のチャンバーに分割され、各チャンバーは、別の色のインクと、内部スポンジと、を収容する。各チャンバーは、第1の端にインク出口を含み、第2の端に通気孔を含む。シーリングテープ1020のカバーは通気チャンネルを覆うために設けられ、ラバーシール1023はインクチャンバーの第2の端をシールするため設けられる。

20

【1610】

内部インクチャンバースポンジ、及び疎水性チャンネルは、プリントロールを移動可能な環境、多種多様な向きでも利用できるようにさせる。更に、スポンジは、それ自体に疎水化処理をすることができるので、インクをコア部から順序正しく追い出すことができる。

【1611】

一連のリブ（例えば、1027）は、コア部1007とプリントロールフォーマー1008との間の摩擦的接触を最低限にさせるため、コア部の表面に設けることができる。

30

【1612】

プリントロールの殆どの部分は、射出成形プラスチックから構成することが可能であり、プリントロールは、大きい内部インク貯蔵容量をもつ。簡単化された構造は、用紙のカールを取り除くメカニズムと、漏れを最小限に抑えるインクチャンバー通気孔と、を含む。ラバーシールは、インク供給チャンバーとの間に効率的な通路を設けるので、動作的能力が高まる。

【1613】

Art cardは、勿論、それ以外の様々な環境で使用される。例えば、Art cardは、組み込み型及びパーソナルコンピュータ（PC）アプリケーションの両方で使用可能であり、大量のデータ又はコンフィギュレーション情報への使い勝手のよいインタフェースを提供する。

40

【1614】

この結果として、多数のアプリケーションが見込まれる。例えば、Art CardリーダーはPCに取り付けてもよい。PCのアプリケーションは多種多様である。最も簡単なアプリケーションは、低コスト読み出し専用配布媒体である。Art Cardは印刷されるので、社内のデータ配布のため使用されるならば、監査の証跡を与える。

【1615】

更に、PCは、しばしば、クローズドシステムのベースとして使用されるが、多数のコンフィギュレーション上のオプションが存在する。複雑なオペレーティングシステムのユー

50

ザ用インタフェースに頼るのではなく、A r t C a r dをA r t C a r dリーダーに単に挿入するだけで、あらゆるコンフィギュレーション上の必要条件を満たすことができる。

【1616】

A r t C a r dの裏面は、アプリケーションとは無関係に、同一の視覚的外観を備えているが（なぜならば、データを格納しているからである）、A r t C a r dの表面はアプリケーションに依存している。表面は、アプリケーションの状況においてユーザに理解される。

【1617】

したがって、図Z35の装置は、本、新聞、雑誌、技術マニュアル等の形で情報の効率的な配信を行う。

【1618】

更なるアプリケーションでは、図Z36に示されるように、A r t c a r d 80の表面は、サンプル画像に適用される芸術的な効果を含む画像を表示する。カード80の裏面にプログラムされたデータを読み出すカードリーダー82を含むカメラシステム81を準備し、アルゴリズム的なデータをサンプル画像83に適用して出力画像84を生成することが可能である。カメラユニット81は、オンボードインクジェットプリンタと、サンプル画像データを処理するために十分な処理手段と、を含む。A r t C a r dの構想の更なるアプリケーションは、B i z C a r dと呼ぶものであり、会社情報を名刺に記憶させる。B i z C a r dは、会社情報の新しいコンセプトである。b i z C a r dの表面は、図Z37に示されるように、外観と機能の点で現在の普通の名刺と全く同じである。それは、写真と連絡先情報とを含み、名刺と同じように多種多様なカードスタイルがある。しかし、各b i z C a r dの裏面は、黒色ドットと白色ドットの印刷された配列を含み、会社に関する1から2メガバイトの情報を保持する。その結果は、3.5インチ形ディスクットの記憶装置が各名刺に取り付けられていることと同じである。

【1619】

情報は、会社情報、特定製品シート、ウェブサイトポインタ、電子メールアドレス、経歴等、b i z C a r dの所有者が望む情報であれば何でも構わない。B i z C a r dは、U S Bポートを用いて標準的なP Cに接続可能なP C付属型カードリーダーのような如何なるA r t C a r dリーダーでも読むことが可能である。B i z C a r dは、ドキュメントとして専用組み込み装置に表示させてもよい。P Cの場合、ユーザはそのb i z C a r dをリーダーに挿入するだけでよい。b i z C a r dは、好ましくは、ウェブサイトと全く同様に、通常のウェブブラウザを使用してナビゲートされる。

【1620】

所有者の写真及びデジタル署名を、会社の公開鍵へのポインタと共に収容することによって、各b i z C a r dは、その人が、本当にその人本人であり、且つ指定の会社のために実際に働いている人であることを電子的に検証するために使用できる。その上、会社の公開鍵を指すことにより、b i z C a r dは、安全な通信を簡単に開始させることができる。

【1621】

T o u r C a r dと呼ばれる更なるアプリケーションは、ある都市への旅行者及び訪問者のための情報を格納したA r t C a r dのアプリケーションである。t o u r C a r dがA r t c a r dブックリーダーに挿入されると、情報は次の形式をとり得る：

＊地図

＊公共輸送機関時刻表

＊名所

＊郷土史

＊行事及び催事

＊レストラン案内

＊ショッピングセンター

T o u r C a r dは、観光客用パンフレット、ガイドブック、及び街角案内の低コスト代

10

20

30

40

50

替物である。製造コストがカード1枚当たりわずか1セントであれば、t o u r C a r dは、観光客案内所、ホテル、及び観光地で、最低限のコスト、又は広告による資金援助があれば無料で配布可能である。ブックリーダーの携帯性は、これを旅行者にとって完璧なソリューションにする。T o u r C a r dは、A r t C a r dリーダーを備わっているコンピュータが任意のウェブブラウザ上でt o u r C a r dに符号化された情報を復号化することができる情報キオスクでも使用可能である。

【1622】

ブックリーダーの双方向性は、t o u r C a r dを非常に多用途化する。例えば、地図に格納されたハイパーテキストリンクは、呼び物ビルディングの歴史的物語を表示するために選択可能である。このように、旅行者は、その都市のガイド付きツアーに出かけることが可能であり、関連した交通機関の路線図及び時刻表は何時でも利用可能である。t o u r C a r dは、地図と、ガイドブックと、時刻表と、レストラン案内と、を別々にする必要性を無くし、各個の旅行者に対する簡単なソリューションを作成する。勿論、データカードにはこの他にも多くの利用可能性がある。例えば、新聞、学習案内、ポップグループカード、ベースボールカード、時刻表、音楽データファイル、製品パーツ、広告、テレビガイド、映画ガイド、見本市情報、雑誌のクーポンカード、レシピ、求人広告、医療情報、プログラム及びソフトウェア、競馬新聞、電子新聞、年次報告書、レストランガイド、ホテルガイド、バケーションガイド、翻訳プログラム、ゴルフコース情報、ニュース放送、コミック、気象詳細等がある。

【1623】

例えば、A r t C a r dは、本の内容、又は新聞の内容を含み得る。このようなシステムの一例は図Z35に示され、A r t C a r d 70は、一方の面に本のタイトルを含み、もう一方の面には本の符号化された内容が印刷されている。カード70はリーダー72に挿入され、リーダー72には、カードリーダー72の折り畳みを可能にさせるフレキシブルディスプレイ73が含まれ得る。カードリーダー72は表示制御部74を含み、表示制御部は、前ページへの移動、次ページへの移動、及びカードリーダー72のその他の制御を可能にさせる。

アートカードの特別の使用

技術説明

技術04の説明

【1624】

さらなる改善では、アートカム装置は、マイクロフォン装置や関連の録音技術を装備するように適当に変更される。写真が撮られる時、周囲の音環境か、その画像に関係するメッセージのいずれかを録音する機会が与えられる。そして、録音された音声は、エンコード形式により出力写真の裏面に印刷される。そのエンコード化は高い復元性を有することが望ましい。その録音された音声は写真に関係する、永久的な音声録音を提供する。次に、再生装置は、エンコードされた音声をスキャンし、この情報を復号化するために提供される。

【1625】

ここで、図233を参照すると、アートカム配列のさらなる改善1601が概略形式で示されており、該配列では、画像1602がCCDセンサー1603により検知され、重要な計算リソースを有するアートカム中央プロセッサ1604に転送されるようになっている。該アートカム中央プロセッサ1604は、メモリー1605に画像を格納でき、該メモリー1605は、好ましくは、メモリーに結合された高速RAMBUS（登録商標）を備える。アートカム中央プロセッサ1604は、また、要求に応じてインスタント写真を提供するため、フルカラー写真、例えば1607の印刷のためにプリントヘッド1606の駆動を制御するようになっている。

【1626】

さらなる改善では、そのカメラ編成1601はサウンドチップ1610も備えており、該サウンドチップ1610は、ACPプロセッサ1604の制御下で、RAMBUS16

11を介してメモリー1605に接続されている。そのサウンドチップ1610は、標準的又は専用形式であることができ、例えばサウンドマイクロフォン1613からアナログ入力1612を取り込むDSPプロセッサを備えることができる。或いは、チップの複雑度が増大するのに従い（ムーアの法則）、サウンドチップ1610の機能はACPチップ1604内に組込まれることができ、ACPチップ1604は、最先端のCMOS型集積回路チップを備えるのが好ましい。本発明の範囲内に該当する他の多くの種類の編成を提供できることは容易に明白であろう。

【1627】

サウンドチップ1610は、アナログ入力1612を対応するデジタル形式へ変換し、それをメモリー1605に格納するために転送する。録音プロセスは、カメラ装置上のボタン（図示せず）を押すことによって作動させることができ、そうでなければ、ボタンがACPプロセッサ1604の制御下であって、録音プロセスは写真を撮るときに実質的に自動化されることができる。録音されたデータは、メモリー1605に格納される。

10

【1628】

図234を参照すると、カメラ編成は、好ましくは、2つのプリントヘッド1615、1616からなるプリンター装置1606を有し、第1のプリントヘッド1615はプリントメディア1617に画像を印刷するのに使用され、第2のプリントヘッド1616はプリントメディアの裏面に情報を印刷するのに使用される。

【1629】

図235を参照すると、写真メディア1617の裏面への見本の出力が示されている。該出力は図234のプリントヘッド1616にて印刷されたものである。その情報は、場所や日付や時間のデータ1620を含むことができる。そして、場所のデータは、キーボード入力により、或いは、GPSや同様のもののような、付属の位置決めシステムの利用によって与えられる。それらの情報1620は、印刷された画像の記録をユーザが整備するために、視認可能な形式で呈示される。重要なことは、画像1617の裏面には、エンコード形式の音声情報1622も印刷されているということである。エンコード形式は多種多様であることができるが、擦り傷、汚れ、書き込み、磨耗、回転、退色などを許容するよう、エンコード化は高い障害許容力を有する方法で行われるのが好ましい。図236には、データ1625の一部分が概略形式で示され、そのデータは、図234のプリントヘッド1616により印刷されるドット列を有する。その開示された手法は写真の裏面に約2.5メガバイトの任意のデータを保管し、この特別な場合には、音声データを含めることができる。符号化フォーマットは、データのリード・ソロモン符号化の利用に強く依存し、高い障害許容力を提供する。さらに、画像の検知を補助するよう、図236に示されるように、高周波“チェッカーボード状”模様がデータに追加されるように記載される。図234のプリントヘッドにより印刷されるドット列をデータが含むようなデータ部1625が図236に概略形式で示される。

20

30

【1630】

録音された音声を“再生”したいときは、読み取り装置1626を通り抜けるように写真1617を移動させる。その読み取り装置1626はピンチローラーを有しており、該ピンチローラーは写真を挟んだ状態で回転し、CCDリニアセンサー装置1627を該写真が通過するようにする。

40

【1631】

図238を参照すると、図237の音声読み取り装置1626の操作を示す概略形式が示されている。前記CCDリニアセンサー1627は第2のアートカム中央プロセッサ1628に接続されており、該プロセッサ1628は、写真の裏面に格納されているデータを読み取り復号化するよう適切に適合化されている。復号化された音声情報は、サウンド処理チップ1633を介してスピーカー1629で再生するためにメモリー1632に格納される。そのサウンド処理チップ1633はACPデコーダ1628の制御下で動作でき、ACPデコーダ1628は、音量制御、巻戻し、再生および早送り制御などを含むことのできる、種々のユーザ入力制御1633の下で動作する。

50

【1632】

重要なことには、上述の通り、アートカードの裏面にあたかも情報が印刷されているかのように、CCDリニアセンサー1627及びACPデコーダ1628は読み取りプロセスを実行する。

【1633】

好適な実施の形態の上記の説明から分るように、写真と関係する音声録音を提供するように、出力画像と関係する音声の自動録音のためのシステムが提供される。写真の裏面の画像出力を読み取るための音声読み取りシステムも開示される。

技術06の説明

【1634】

さらなる改善では、先述したアートカムは、後のデジタル画像の生成に利用されるカメラの向きを自動的に決定するための装置を含む。そして、そのアートカムは、画像生成に向きの情報を利用することを適切にプログラムされている。

【1635】

ところで、向きの情報を使用する多くの例が添付図面を参照して説明される。図239を参照すると、第1のポートレート写真1801が示されており、該写真1801には、カメラシステムに入力された日付けや撮影場所情報のような、ポートレート1801に関連した情報を配置するために領域1802が用意されている。さらに、そのポートレート1801は、顔などのような、画像の特定の目に見える部分を変形させる画像処理技術により処理されるかも知れないと予想される。これらの変形は、写真1801を撮影するカメラの向き決定に依存する。また、図239には、カメラが回転されたポートレートモードで使用された第2のポートレートモード画像1803が示されている。残念なことに、横向きモードの画像1801の画像データ1802が同じ位置にあるならば、該データは、最後の画像における望まれない領域1805に現われる。理想的には、ポートレートモードで写真が撮影されるとき、画像の特定の情報は場所1806に現われるべきであろう。なお、直角で、異方性で、画像内容の特定の方法で画像に与えられる画像処理効果は、画像が回転されるときに無駄になってしまうだろう。

【1636】

前記向きセンサー1846は向きを検知し、該検知された向きはACP1831によって、対応するデジタル向き角度に変換される。前記エリアイメージセンサー1802はカメラにより撮影される画像を検知し、該画像はACP1831に転送され、メモリー記憶装置1833に保存される。次に、検知された向きは、撮影された画像がポートレートモードか風景モードであるかを決定するために使用される。該検知された画像はメモリー記憶装置1833から読み取られ、再びメモリー記憶装置1833に出力される前に回転される。最後の結果が回転された画像1808である。また、検知された向きの情報を利用するための他の技術を挙げることができる。例えば、メモリー記憶装置1833に保存されている検知画像は、向き情報と関連付けて保存されることができ、該向き情報は、従来の回転マトリクスマッピング、世界座標系及び局所座標系マッピングに利用される。続いて、その情報の処理を行うコンピュータ・グラフィック・アルゴリズムは、回転向きに画像を操作するように処理している最中にその座標系を使用することができる。位置検知情報（例えば、図239の符号1806）が画像に組み入れられる場合には、このテキストは正しい向きに配置されることができる。さらに、カメラの向きは、顔検出のような、画像に適用される画像を特定するアルゴリズム技術に使用されることができる。顔の位置は大体分るので、向きの情報の利用は顔検出アルゴリズムを大幅に単純化する。

【1637】

向きセンサーは、例えばマーキュリーセンサのような、単純で安価な重力スイッチを含むことができ、或いは、マイクロマシンシリコン加速度計のような、より複雑な小規模の装置にすることができる。

技術07の説明

【1638】

さらに、プリントローラーの他の形態が構成されることができる。図 2 4 1 を参照すると、前述した装置の修正版に利用されるようなプリントロールの他の配列が概略的に示されている。重要なことには、使い捨てのプリントロール（例えば、1 9 1 0）は、インクを集めることを許容するスポンジ部 1 9 1 2 を有する“延長されたリーフ部分 1 9 1 1”を備える。従って、プリントヘッドを横切るように搬送されるフィルムや紙の端縁部へのプリントヘッドによる印刷を許容するため、運転中、前記スポンジ部 1 9 1 2 はプリントヘッドの下方に配置される。

【1 6 3 9】

そのスポンジ部 1 9 1 2 は、使い捨てのプリントロール 1 9 1 0 の一部を形成するので、プリントロールを使い切ったときにカメラの外に取り出され、それによって、余分の不
10

技術 0 8 の説明

【1 6 4 0】

さらなる実施例では、最大のコントラストを確保する C C D データストリームの処理によりオートフォーカスが実行される。C C D データストリームに基づき焦点位置を決定する技術は公知である。例えば、Butterworth-Heinemann によって 1 9 9 3 年に出版され、Leslie Stroebel と Richard Zakia により編集された“The Encyclopedia of Photography”や、“Applied Photographic Optics（ロンドン及びボストン、フォーカルプレス、1 9 8 8 年）”にて言及されている。これらの技術は、第一に、入
20

【1 6 4 1】

このオートフォーカスの情報は、特定のモード、例えば、画像の中に顔の位置を見つけようとする場合に A C P によって使用され、画像における顔の大体のサイズを案内することによって顔位置検知プロセスを簡素化している。

【1 6 4 2】

図 2 4 2 を参照すると、顔検出アルゴリズムによる調査のために大体の画像の特徴を決定すべく使用される方法の見本 2 0 0 1 が示されている。

【1 6 4 3】

例えば、2 0 0 2，2 0 0 3，2 0 0 4 のような様々な画像がカメラ装置により取得される。焦点設定の詳細は、オートフォーカスの操作の副産物として A C P に保存される。加えて、ズーム・モータの現在位置もまた使用される（2 0 0 6）。これらの設定のいずれもが A C P により決定される。次に、大体の奥行き位置（depth location）に応じた大きさとなる、画像中のオブジェクト（興味のあるオブジェクト）を含む画像を出力する前に、A C P は、検知した値に分析技術 2 0 0 8 を適用する。
30

【1 6 4 4】

次に、画像におけるオブジェクトの位置を見つけるため、該奥行きの値（depth value）が、入力された検知画像 2 0 1 1 に加えて A C P 2 0 3 1 の顔検出アルゴリズム 2 0 1 0 に利用される。近距離値 2 0 0 9 はポートレート画像の高い可能性を示し、中距離値は
40

技術 0 9 の説明

【1 6 4 5】

さらなる実施例では、アートカム装置は、そのときのアイポジションを検知するアイポジションセンサーを有するように変更されることができる。その検知されたアイポジション情報は、その検知されたアイポジションに従った修正や変形等を行うため、カメラによ
50

り撮影されたデジタル画像を処理するために使用される。

【1646】

アイポジションセンサーの構成は当業者に知られており、多くの製品のカメラ（特に、キヤノン社のカメラ）にて使用されている。アイポジションセンサーは、ビューファインダから視聴者の目への赤外線照射と、大体のアイポジションを決定するために検出され使用される反射とに依存している。

【1647】

図243を参照すると、アイポジション情報2110と画像2111とはアートカムのメモリーに格納されており、ACPによって処理が行われ（2112）、その処理画像は、プリントヘッドにより写真として印刷するために出力される。画像処理2112の形態は、アイポジション情報2110に依存してかなり変化されることができる。例えば、画像処理の第1の形態においては、画像中の顔の位置を検出して、様々なグラフィック・オブジェクト（例えば、顔との関係で一般的にオフセットしている吹き出し）を適用するために、顔検出アルゴリズムが画像2111に適用される。そのようなプロセスの一例が図244に示されており、第1の画像2115は3人の人間を示している。前記顔検出アルゴリズムの適用により、3つの顔2116、2117、2118が検出される。そして、アイポジション情報は、フレーム中に推定された目に最も近い顔選択するために使用される。第1の例においては、吹き出しは頭2116に対して配置される。第2の例2120においては、吹き出しは頭2117に対して配置され、第3の例2121においては、吹き出しは頭2118に対して配置されている。したがって、画像中の所定の顔の上方に吹き出し文字を配置するために、アートカードは、吹き出し適用アルゴリズムのコード化された形態と、処理された画像とを含むように提供される。

【1648】

アイポジション情報は多くの異なる方法で画像2111を処理するために使用することができるということは容易に理解される。局所的な、或いは特定の方法で焦点合わせの効果を適用し、顔やオブジェクトに特定の変形を与えることができる。さらに、行われた画像処理は、画像の芸術的な演出を含み、該演出は、芸術的な絵筆技術を適用することを含む。その芸術的なブラッシング方法は、アイポジション情報2110に従って特定の方法で適用され得る。最終的な処理画像2113は、要求に応じて印刷され得る。さらなる画像が撮影され、異なる出力画像の生成のため、異なるアイポジションが検出され使用される。

技術10の説明

【1649】

さらなる改善では、アートカムは、撮影画像の明るさのレベルを決定するために自動露出センサーを有するように構成変更されている。この自動露出センサーは、画像の一部分を引き立たせるため、設定された露光量に従って画像を処理するために使用される。

【1650】

望ましくは、前記エリアイメージセンサーは、画像を撮影するときの光の状態を決定するための装置を有する。該エリアイメージセンサーは、検出したレベルに従って、CCDにより取得されたダイナミックレンジの値を調整する。取得した画像はアートカム中央プロセッサに送られ、メモリーストアに格納される。前記エリアイメージセンサーにより測定された強度情報は、また、ACPに転送される。この情報は、保存されている画像に特定の効果を与えるため、ACPによって使用される。

【1651】

図245を参照すると、ACPによって画像を処理するため、自動露出設定情報2201が、保存されている画像2202に関連して使用される。処理された画像は、後でプリンターで印刷2204するためにメモリーに戻される。

【1652】

所定の光の状態に従って多数の処理ステップが実施されることができる。暗い場所で画像が撮影されたことを自動露出設定情報2201が示す場合には、画像の色をより強く、

より深く、より鮮やかにするために、画像の画素の色が選択的にリマップされる。

【1653】

写真が撮影されるときが明るい状態であると自動露出情報が示す場合には、より明るく、よりサチュレートするように画像の色が処理される。画像の再配色は、色相・彩度・明度（HSV）フォーマットに画像を変換し、要求に従ってピクセル値を変更することにより行うことができる。該ピクセル値は、印刷の出力色フォーマットに変換して出力することができる。

【1654】

もちろん、数多くの異なる再配色技術が使用されるかも知れない。それらの技術は、好ましくは、読み出し装置に挿入されるアートカードに明確に示されている。他の方法としては、ある状態での使用のため、画像処理アルゴリズムが自動的に適用され、カメラの中に組み込まれることができる。

【1655】

他の方法としては、挿入されたアートカードは、画像に適用される、自動露出設定に限定された、多くの取扱いを有することができる。例えば、暗い画像には、蠟燭等を含むクリップ・アートが挿入されることができ、明るい画像には大きな太陽が挿入されることができる。

技術11の説明

【1656】

フラッシュを使用した結果、取得画像に生じる“赤目”の影響を除去することは、重要な処理の1つである。図246を参照すると、さらなる改善において、CCD装置により取得された画像は、フラッシュが使用されていた場合には、印刷のために処理された出力画像を形成するため、処理ステップ3を受けさせる。図247を参照すると、CCD装置による画像の取得の際にフラッシュが使用されていた場合に使用されることのできる、1つの特別な画像処理アルゴリズム2310が、より詳細に示されている。そのアルゴリズムは、好ましくは、CCDにより画像を取得するためにフラッシュが使用されたときにのみ使用される（2311）。そのアルゴリズムの目的は、フラッシュの利用による画像への影響を低減することである。そのような画像の影響は、写真画像において個々の眼が赤く見えるという、周知の“赤目”影響を含むことができる。また、フラッシュ光の反射面からの反射などのような他の影響は、別途、他のアルゴリズムを使用して処理することができる。画像における赤目の影響を排除するための第1ステップは、画像中における顔を決定することである（2312）。何らかの他の画質増強や色修正を行った後、通常光の下で人間の顔の色についてのHSVの値の地図で近接する範囲を決定することにより、顔検出プロセスが実行される。検出された範囲については、目や口や全体形状や重なり の決定を含む、様々な簡単なテストがなされる。その簡単なテストは、顔の可能性についての結果を生じさせることができ、閾値以上の場合には、画像での顔の位置を決定することができる。

【1657】

画像の中で顔が決定されると（2312）、その顔の中で眼の位置が見つけれられる（2313）。そして、赤目除去処理が必要かどうかを決定するため、特別な範囲の色を判断する処理がそれぞれの眼に個別に行われる。もし、赤目除去処理が必要とされるならば、出力画像において、如何なる不連続さや起こり得る不自然さを生じないようにさせながら、赤色を減少させるようにすべく、目のエリアに対して修正アルゴリズムが適用される。もちろん、眼の中心点の周りのガウシアン・デグリー の形式を含む多くの異なる技術が使用される。最後に、メモリーストア2333の裏にアップデートされた形式で画像が書き込まれる（2316）。

【1658】

望ましくは、この時点で、フラッシュ光のスペクトル特性の影響を受けた色をリマップすることを含む、他の修正アルゴリズムが使用される。代わりに、挿入されるアートカードが、画像に適用される多くの処理（フラッシュの設定を特定する処理）を行うようにし

10

20

30

40

50

ても良い。例えば、蝋燭や光の球体 (light globe) 等のクリップアートが、如何なる閃光も無い画像中に閃光や大きな太陽が挿入されても良い。

技術 12 の説明

【1659】

さらなる改善においては、第2のインクジェットが写真の裏面への印刷のために用意されている。図248には、標準的な配列の概略図が示されている。ローラー（例えば、2404）に挟まれる紙2403に写真を印刷するためにプリントヘッド2402が用意されている。2つのインクジェット・プリントヘッド2402、2411を備えるという実施例に従い、図249には代替案2410が示されている。プリントヘッド2411は、写真に関連のある重要な情報を写真2403の裏面に印刷するために用意されている。例えば、その写真を撮った時間や日付が、そのインクジェット・プリントヘッド2411によってその裏面に印刷されることができる。加えて、そのアートカム装置はグローバル・ポジショニング・センサーを備えるかも知れない。該センサーは、アートカム中央プロセッサによってデータ取り出しが行われ、その写真が撮られた場所を確定するために利用されるかも知れない。この付加情報は、また、写真出力2403の裏面に印刷されるかも知れない。

技術 13 の説明

【1660】

プリントロールは堅く巻かれた形態で提供される。残念なことに、プリントロールを堅く巻いた結果として、プリントメディアの特性が、アートカムシステムにより出力される写真には過量のカールがもたらされるというようになるかも知れない。

【1661】

図250を参照すると、3個のローラー配列2511-2513を有するシステムを提供する、プリントロールの別の配列が示されている。そのローラー配列は、プリントロールを抜け出たプリントメディア2514に、しっかりとしたアンチカールを与える。それらの3つのローラー2511-2513はメディア2514を挟み込み、ピンチローラー（例えば、2517）によってプリントヘッド2516へ搬送される前にその紙にアンチカールを与える。それらの3つのローラー2511-2513は、プリントロールケース、或いはアートカム装置の内部に配置されることができる。

技術 15 の説明

【1662】

図251を参照すると、さらなる改善では、観察者2703に、立体的、或いは疑似三次元的な効果で見せるような写真画像を作ることが望まれる。これらの効果は、2つの画像をすぐ近くに同時に記録し、1つ目の画像を観察者の右眼2704に与え、2つ目の画像を観察者の左眼2705に与えることにより得ることができる。左眼と右眼のために画像を記録し、そして左眼には左の立体的画像を見せ、右眼には右の立体的画像を見せることにより、3次元の立体効果を作り出すことができる。

【1663】

さらなる改善では、左右の眼の画像を取得する、一連のレンズ状の透明な柱によって、写真や立体画像2702は、以下のように、よりはっきりとなるように、構成されるだろう。

【1664】

図252を参照すると、写真紙2702の表面部分の断面図が示されている。断面図2710に明らかなように、写真紙は、左右の立体視が可能となるようにデザインされた、一連のレンズ状の列を有する。画像は、透明な写真紙2710の底面2712に印刷される。

【1665】

1つの画素のピッチ2713は80 μm で、ドットの配列ピッチ2714は20 μm である。それ故に、1つの画素につき4つのドット2714-2717が配置される。2つのドット2714-2715は左側の立体視のために用意され、2つのドット2716-

10

20

30

40

50

2717は右側の立体視のために用意される。その写真紙を見る右眼は右側の画素ドット2716, 2717の像を取得し(2720, 2721)、左眼は左側のドット2725, 2726の像を取得する(2723, 2724)。それ故、2つの画像がそれぞれの眼に別々に示され、実体写真効果が生じる。主として、2711のような各列(column)のレンズ状の外形に起因する実体写真効果。

【1666】

ここで、図253を参照すると、紙2710の下面部の斜視図が示されている。図253の説明図は、4つのドット2731-2734を有する画素のピッチに加えて、各ドット2730のドットピッチを示すために作図線を含む。図253に明らかなように、右側立体視のために用意された2つのドット2731-2732、及び、左側立体視のために用意された2つのドット2733-2734の4つのドットにより各画素が構成される。そのドット、例えば、2730はプリントメディア2710に配置されているので、該メディアが裏返された時に、正しい立体視が行われる。

10

【1667】

ここで、図254を参照すると、立体画像を撮像する1つの方法が論じられるだろう。望ましくは、携帯用カメラ装置に配置されたCCDカメラ2740, 2741により2つの画像が撮像される。各CCD装置2740, 2741は別々の画像を取得し、それら別々の画像は、図235に示されるような左右交互の方法で、立体処理ユニット2742にて結合されるべきである。画像が立体処理ユニット2742にて正しく交互配置されると、その画像は、プリントエンジン2743、及び、写真紙2710の表面2746にその画像をプリントするプリントヘッド2744に送られる。

20

【1668】

写真紙2710の現在位置が位置合わせユニット2750により検知され、その現在位置が、プリントヘッド2744の制御のためにプリントエンジン2743によりフィードバックされる。その位置は、LEDタイプの装置2780がプリントメディア2710のレンズ状面を照らすと共に、プリントメディア2710の反対側の光導電体2781が周期的な明るさの変化を測定することにより決定される。

【1669】

図255を参照すると、破線2750により示される位置合わせユニットの動作がここでさらに説明される。写真紙2710はローラー2751, 2752に挟まれ、ローラー2752は、写真紙(プリントメディア)2710のレンズ状面2754に一致する面を有するように構成されている。その写真紙2710のレンズの周期が2780mmのオーダーなので、ローラー2752の表面は、写真紙2710のレンズ状面のよう、誇張して示される。レンズ状写真紙の緻密な周期は、ローラー2752がプリントメディアの他の形態にも同様に使用できることを意味している。

30

【1670】

ローラー2752の表面は、ローラー2752との一定の空間的關係をプリントメディア2710が維持することを保証する。従って、要求に応じてローラー2752を回転し、それによってプリントヘッド2744を通過するようにプリントメディア2710を搬送するように、プリントローラー2751, 2752はプリントエンジン2743の制御下で位置合わせユニット2750によりモニターされながら駆動される。結果として、プリントヘッド2744により印刷されるドットとレンズ状列との位置の合致が維持される。

40

【1671】

プリントメディア2710がプリントヘッドを通過するように搬送される際に、プリントヘッド2744により立体画像をプリントすることができる。プリントヘッド2744としては、フルページ幅のプリントヘッドが望ましい。

【1672】

もちろん、プリントヘッド2744に関連する、プリントメディア2710の現在位置を他の装置により決定することも可能である。

50

【1673】

図256を参照すると、好適な形状のレンズ状紙2761と、対応するインク供給（不図示）とを有するプリンターロール2760を結合することにより、カメラ装置が、提案された実施例の原則に従って提供され得る。その紙2761は、プリントローラー2760からピンチローラー2763、2763によってプリント装置2764に取り出される。該プリント装置2764は、さらに、ピンチローラー2765、2766、カッター2767、プラテン2769、ピンチローラー2751、2752、及びプリントヘッド2744を有する。立体画像はプリントヘッド2744によってプリントされ、その紙2761がカッター2767により切断された後に取り出される。その出力2770は透明な形態である。

10

【1674】

図257を参照すると、画像の排出物2770の上に、平坦な白の面2771を付着することができる。これは、透明2770を貼り付ける粘着面の使用により成し遂げられる。続いて、左右のステレオ画像を生成して、撮影画像に三次元効果を与えるように、図251に示されるように、レンズシステムを通して画像が視認される。図256に描かれたカメラシステムは、理想的には、写真を撮影して直ぐに立体写真を得たい場所に運ぶことができるような携帯用である。

技術17の説明

技術21の説明

【1675】

さらなる改善では、堅くロール状に巻かれて格納される画像メディアは、異方性のリブ構造を持つように処理され、ロール状で運ぶのに好適で、該プリントメディアに印刷された画像を見るときには該プリントメディアのカール度合いが減少されるようになっている。

20

【1676】

ここで図259を参照すると、この改善の本質に従って取り扱われるプリントメディア3311が示されている（3310）。該プリントメディア3311は、他の形態が好適だけれども、プラスチックの平面的なフィルムにて構成される。該プリントメディア3311は、図259の拡大図に示されるように、ポリマーのリブ状構造3312を有するように前処理されており、実際の列のピッチが約200mである。したがって、該プリントメディア331の一方の表面は、該プリントメディア331の全長を上下に走行する一連の列3312を有するように処理されている。該列3312の断面は図に拡大図で示されている。

30

【1677】

一連の列3312を使用することのメリットは、プリントメディア3311の表面部分に力3313が加えられた場合に明らかである。その列3312は、プリントメディア3311が堅く巻かれることを許容する一方で、3316の向きにプリントメディア3311が巻かれることに抵抗する。従って、加えられた如何なる力3313も、3315及び3316の両向きに伝達しそうである。そのメディア3311の異方性の性質は、3315及び3316の両向きに支持を与えるように機能するリブ3312によってもたらされ、それによってメディア3311のカールを制限する。また、その材料の異方性の強度は、ロールの中心軸に沿った強度で格納されることが許容される。

40

【1678】

図260を参照すると、ユーザの手3320、3321の中で画像3311を単に保持して、ポイント3324、3325に僅かな圧力を与えることによって、カーリングによる効果を減少させて、メディア3310上の画像を見ることができる。

【1679】

もちろん、異方性のメディアは、該メディアがプラスチック材料を含んでいる場合には多くの技術を利用して製造されることができる。例えば、それは、押し出し成形法により製造されることができる。或いは、他の技術が利用され得る。例えば、1つの製造方法が

50

図 2 6 1 に模式的に示される (3 3 4 0)。該方法では、ローラー 3 3 4 2, 3 3 4 3 の間でメディア 3 3 4 1 がプレスされるようになっている。図 2 6 1 (a) に示すように、ローラー 3 3 4 3 はフラットな面を有し、ローラー 3 3 4 2 はギザギザのある面 3 3 4 5 を有している。説明のため、鋸状の外形は図 2 6 1 に拡大されている。

【1 6 8 0】

メディアがそのようなプロセス 3 3 4 0 の利用に適していない場合、異方性メディアは、互いに貼り合わされた 2 枚のフィルムの面を利用することによってもたらされる。そのようなプロセス 3 3 5 0 は図 2 6 2 に示される。画像の印刷が望まれる第 1 面 3 3 5 1 は、接着剤や熱融合等の方法によって要求通りに第 2 面 3 3 5 2 に貼り付けられる。

【1 6 8 1】

さらに、プリントメディアの製造の他の方法が可能である。例えば、図 2 6 3 及び図 2 6 4 を参照すると、プリントメディア 3 3 6 0 の構造が断面図で示される。該プリントメディア 3 3 6 0 の構造は、繊維型の中に引かれる、軟ポリエチレン・ナフタレートにより形成された繊維からなる強化ポリマーを含む第 1 繊維材料 3 3 6 1 から形成される。ポリエチレン等を含む第 2 の熱流動ポリマー (heat-flowable-polymer) 3 3 6 2 は、インク／画像の化学薬品のキャリッジのための“紙”の基材を提供するために使用される。繊維 3 3 6 1 と印刷ベース 3 3 6 2 とが、図 2 6 4 に示すように一体ユニットを形成するように、メディア・ポリマー 3 3 6 2 は粘着力が残っている間 (例えば、加熱後) 繊維材料 3 3 6 1 に押し付けられる。図 2 6 3 及び図 2 6 4 のメディアの配列は、多くの技術に従って作られることができる。次に、図 2 6 5 を参照すると、繊維状材料の成型スプール 3 3 8 0 を使用する、第 1 のそのような技術が示されている。その技術では、多くのスプールと、対応する繊維 3 3 8 1 とが、所望のプリントロール長に相当している。それらのスプール 3 3 8 0 は、対応する繊維 3 3 8 1 が第 1 ロール 3 3 8 2 に供給されるように配列されている。そして、それらの繊維 3 3 8 1 は、加熱された粘着性のあるプリントメディアの層を該繊維に供給するプリントメディア塗布ユニット 3 3 8 4 を通過するように引かれる。その後、ローラー 3 3 8 5, 3 3 8 6 が、繊維とプリントメディアとが一つに融合するように必要な圧力を供給すると共に、最後に異方性面 3 3 8 8 を形成するようにプリントメディアの面を平らにする。その後、該面 3 3 8 8 は裁断され、上述のように異方性プリントメディアを形成するように巻かれる。

【1 6 8 2】

もちろん、要望に応じて繊維状材料の形成を他の形態で行うことが可能である。例えば、図 2 6 6 には、タンク 3 3 9 0 からファイバ 3 3 9 1 を引く方法により繊維状材料を押し出して生成する 1 つの形態が示されている。図 2 6 6 の装置は図 2 6 5 のスプール 3 3 8 0 に置き換えることができる。

技術 2 4 の説明

【1 6 8 3】

さらなる改良版では、前記アートカムシステムに利用されるアルゴリズムが説明される。該アルゴリズムは、入力画像中における一群の画素を、出力画像では“ブラッシュストローク”で置き換えることにより、写真画像を“描いた”ような演出に自動変換するものである。該アルゴリズムは、ブラッシュストロークをヴァン・ゴッホ・スタイルに近付けるように、画像中のフラットな部分での主要なエッジを自動検出し、該エッジの方向の情報を増殖させる (propagate) ように働く。該アルゴリズムは、前述のアートカム装置にて履行するのに好適である。

【1 6 8 4】

まず、図 2 6 7 を参照すると、そのアルゴリズムは多くのステップ 3 6 0 1 を含む。これらのステップは、エッジを検出するために画像のフィルタリングをする第 1 ステップ (3 6 0 2) を含む。次に、それらのエッジは、最終的なエッジを決定する処理 (3 6 0 5) の前に、閾値処理、或いは“スケルトン処理 (3 6 0 4)”される。そして、それらのエッジにベジエ曲線が適合 (fit) される。次に、該曲線はオフセットされ (3 6 0 7)、最終的な画像にブラッシュストロークが描かれる (3 6 0 8)。それらの処理 3 6 0

10

20

30

40

50

7, 3608は、画像がブラッシュストロークにより十分に覆われるようになるまで繰り返される。その後、画像への最後の“修正”が実行される(3609)。

【1685】

次にそれぞれのステップをより詳細に説明する。エッジを検出するためにフィルタリングをする第1のステップ(3602)では、図268に示すような係数の集合3612, 3613を持つソーベル(Sobel)の 3×3 のフィルタが前記画像に適用される。ソーベルフィルタはデジタル画像処理にて使用される、良く知られたフィルタであり、その特性は、マサチューセッツのAddison-Wesley出版社により1992年に出版された、規範となる教科書“デジタル画像処理”にてGonzalez及びWoodsにより197-201頁に完全に討論されている。ソーベル微分フィルタは、フィルタリング前に画像をグレースケールに変換するか、或いは、画像を各カラーチャンネルにフィルタリングして最大値を取ることににより適用されることができる。ソーベルフィルタリングにより、画像の画素単位のエッジ強度を示すグレースケール画像が作成される。

10

【1686】

次に、対応する二値画像を作成するために閾値処理が行われる(3603)。その閾値を変化させることができるが、最大強度値の50%の値が適切である。エッジを強化した画像(edge strength image)中の各画素が該閾値と比較され、もし閾値よりも大きければ“1”が出力され、閾値よりも小さければ“0”が出力される。この処理の結果、閾値エッジマップ(threshold edge map)が作成される。

【1687】

20

次に、図267のステップ3604では前記閾値エッジマップが“スケルトン処理”される。画像をスケルトンする処理は、上述参考教科書の491~494頁や、他の標準的な教科書に完全に明確に記載されている。そのスケルトン処理は、前記閾値エッジマップにおける相当数の特徴を維持しつつ、“薄く”スケルトン化されたエッジマップを作成する。

【1688】

次のステップでは、画像におけるポイントのリストを含むデータ構造を生じさせる、スケルトン化されたエッジマップにおけるエッジが決定される。好ましくは、予め決められた最小値よりも長いエッジのみがリストに加えられる。

【1689】

30

前記スケルトン化された画像は、多重分岐が可能な1画素幅のエッジのみを有するので、C++コード片(code fragment)で表された次のアルゴリズムは、スケルトン化された画像において切れ目のないエッジに属するポイントを決したり同定したりする1つの方法を提示する。それは、分岐したエッジを別々のエッジに分解し、各分岐の湾曲を最小限にする方向にエッジに沿って伸びることを選択する。例えば、分岐ポイントにて、最小の湾曲を生じさせる分岐を助ける。そのコードは以下の通りである。：

```

void
FollowEdges
(
    Image& image,
    int minimumEdgeLength,
    PointListList& pointListList
)
(
    pointListList.Erase();
    for (int row = 0; row < image.Height(); row++)
    {
        for (int col = 0; col < image.Width(); col++)
        {
            If (image[row][col] > 0)
            {
                PointList pointList;

                // append the starting point to the point list,
                // and clear it so we don't find it again
                pointList.Append(Point(col, row));
                image[row][col] = 0;

                // follow the edge from the starting point to its beginning
                FollowEdge(row, col, image, pointList);

                // reverse the order of the points accumulated so far,
                // and follow the edge from the starting point to its end
                pointList.Reverse();
                FollowEdge(row, col, image, pointList);

                // keep the point list only if it's long enough
                if (pointList.Size() >= minimumEdgeLength)
                    pointListList.Append(pointList);
            }
        }
    }
}

```

10

20

30

40

```
// table of row and column offsets to eight surrounding neighbours
// (indexed anti-clockwise, starting east)
static int offsetTable[8][2] =
{
    {0, 1}, {-1, 1}, {-1, 0}, {-1, -1}, {0, -1}, {1, -1}, {1, 0}, {1, 1}
};
```

10

```
// table of preferred neighbour checking orders for given direction
// (indexed anti-clockwise, starting east favouring non diagonals)
static int nextDirTable[8][8] =
{
```



```

    {0, 2, 6, 1, 7, 3, 4, 5),
    {2, 0, 1, 3, 7, 4, 5, 6),
    {2, 4, 0, 3, 1, 5, 6, 7},
    {4, 2, 3, 5, 1, 6, 7, 0),
    {4, 6, 2, 5, 3, 7, 0, 1},
    {6, 4, 5, 7, 3, 0, 1, 2),
    {6, 0, 4, 7, 5, 1, 2, 3},
    {0, 6, 7, 1, 5, 2, 3, 4),
};
10

void
FollowEdge
(
    int row,
    int col,
    Image& image,
    PointList& pointList
)
20
{
    Vector edgeHistory[EDGE_HISTORY_SIZE];
    int historyIndex = 0;

    for (;;)
    {
        // table of pre-computed
        // compute tangent estimate from edge history
        Vector tangent;
        30
        for (int i = 0; i < EDGE_HISTORY_SIZE; i++)
            tangent += edgeHistory[i];

        // determine tangent angle and quantize to eight directions
        // (direction zero corresponds to the range -PI/8 to +PI/8, i.e east)
        double realAngle = tangent.Angle();
        int angle = (int) ((realAngle * 255) / (2 * PI) + 0.5);
        int dir = ((angle - 16 + 256) % 256) / 32;
        40

        // try surrounding pixels, fanning out from preferred
        // (i.e. edge) direction
        int* pNextDir = nextDirTable[dir];

```

```

bool bFound = false;

for (i = 0; i < 8; i++)
{
    // determine row and column offset for current direction
    int rowOffset = offsetTable[dir][0];
    int colOffset = offsetTable[dir][1];

    // done testing neighbours if edge pixel found
    10

    if (image [row + rowOffset] [col + colOffset] > 0)
    {
        // determine edge pixel address
        Point oldPoint (col, row);
        row += rowOffset;
        col += colOffset;
        Point newPoint (col, row);
        20

        // update edge tangent history
        tangent = newPoint - oldPoint;
        tangent.Normalize();
        edgeHistory[histroIndex] = tangent;
        historyIndex = (historyIndex + 1) % EDGE_HISTORY_SIZE;

        // append edge pixel to point list
        pointList.Append(newPoint);
        30

        // clear edge pixel, so we don't find it again
        image[row][col] = 0;
        bFound = true;
        break;
    }
    // determine next direction to try
    dir = pNextDir[i];
}
    40

// done following edge if no edge pixel found
if (!bFound)
    break;
}
}

```

【 1 6 9 0 】

スケルトン化されたエッジマップにこのアルゴリズムを使用することにより、少なくとも所定のサイズを有するエッジのリストが生成される。好適なサイズは、20画素長であ 50

ることが分っている。

【1691】

図267の次のステップ3606では、ステップ3605にて得られた各エッジリストのそれぞれにベジエ曲線が適合される。エッジリストのそれぞれのため、区分的なベジエ曲線がポイントのリストに適合される。区分的ベジエ曲線を適合させる好適なアルゴリズムは、“デジタル化された曲線を自動的に適合させるためのアルゴリズム”シュナイダー、P. J. 著、Glassner, A. S. 編集、Graphics Gems、Academic出版、1990年にて説明されるようなシュナイダー曲線適合アルゴリズムである。このアルゴリズムは、パラメトリックの連続性ではなく幾何学的な連続性のみを目的とし、素早く収束するようにしている。シュナイダーアルゴリズムは機能的であり、もし適合度が乏しければ、最大エラー部で曲線を再分割し、該分割されたものに曲線を適合させる。次に、その分割部分での接線の推定は、分割点のいずれかのみを使用してなされる。密集した点集合のために、ローカルノイズが増幅される傾向がある。曲線を適合することの、改良された品質は、接線の基礎とした分割点から遠ざかる点を使用することで得られる。

10

【1692】

図267の次のステップ3607では、所望の“ブラッシュストローク幅”の半分だけ初期のカーブリストから曲線がオフセットされる。そのオフセットは、最初の主曲線の両側に平行となるように、互いにほぼ1ストローク幅だけ離れる2つの曲線を生じさせる。

【1693】

特定の区分的ベジエ曲線にほぼ平行な区分的ベジエ曲線を生成するため、次のステップを含むアルゴリズムが使用される。

20

i. 空のポイントリストの作成

ii. 空の接線（ベクトル）リストの作成

iii. 区分的ベジエ曲線を形成する曲線部分のそれぞれに選択されたポイント进行评估し、特定のオフセット値によりそれらをオフセットさせる。前記ポイントリストにオフセットポイントを付け足し、対応する接線を前記接線リストに付け足す。

iv. 得られたポイントリストに区分的ベジエ曲線を適合させる。曲線を適合させるプロセスにおいて接線を予測するのではなく、オフセットポイントに関連した正確な接線を使用する。

30

【1694】

各曲線の区分は次のようにオフセットさせる：

i. 曲線値と、標準化された接線と、標準化された法線とを求める。その法線は、0

. 0以上1. 0以下の、等間隔に配置されるパラメータ値（例えば、0. 25の間隔）で画像のサイズが標準化される。

ii. 前記特定のオフセット値により法線をscaleする。

iii. 前記曲線のポイントと縮小された法線とを使用して線分を組み立てる。

iv. もし、2本の線分が交差するならば、1本の線分に関連するポイントを削除する。

。

v. 存続するポイント（surviving point）をポイントリストに付け足し、それらに対応する接線を接線リストに付け足す。もし、問題の線分が区分的曲線の最後のものであったなら、パラメータ値1. 0に関連するポイントのみを付け足し、次の線分のパラメータ値0. 0に関連するポイントは複写する。

40

【1695】

各曲線部分をオフセットさせるプロセスは次のように行われる。すなわち、

1. 初めに、0. 0以上1. 0以下の、ベジエ曲線上に等間隔に配置されるパラメータ値のため、各パラメータ値 P_n （図269）のため、曲線値3630と、標準化された接線3631と、標準化された法線3632とが計算される。

2. 次に、特定のオフセット値により前記法線3632がscaleされる。

3. 次に、ポイント3630からポイント3636（scaleされた法線3634の

50

端のポイント)までの線分が計算される。

4. 次に、その線分3630-3639は、曲線上の他の全てのポイント(例えば、3638, 3639)がチェックされる。もし、いずれか2つの線分が交差しているならば、ポイントの1つ3636は廃棄される。存続するポイント(surviving point)をポイントリストに付け足し、それらに対応する接線を接線リストに付け足す。もし、問題の線分が区分的曲線の最後のものであったなら、パラメータ値1.0に関連するポイントのみを付け足し、次の線分のパラメータ値0.0に関連するポイントは複写する。

【1696】

図270を参照すると、一連のベジエ曲線区分(例えば、C1, C2等)を作成するため、図267のステップ3607に従った曲線のオフセットの最終結果が示されている。最初に、一定間隔で配置された一連のパラメータポイントP1-P5。次に、各ポイントP1-P5に対して、法線のポイントN1-N5(図269のポイント3636に対応する)が作成される。次に、それらのポイントN1-N5を利用して、区分的ベジエ曲線3640が作成される。このプロセスは、ポイントN6-N10を含む反対側の曲線3641にも適用される。このプロセスは、続く区分的曲線C2等にも適用される。この結果、互いに平行で、ほぼ1ブラッシュストロークの幅だけ離れて配置される2本の曲線3640, 3641が得られる。

【1697】

次に、一連のブラッシュストロークが、前記曲線に沿って、出力画像中に配置される。それらのストロークは、曲線の接線方向に方向付けられる。各ブラッシュストロークは、他のブラッシュストロークと重ならない場所を規定するフットプリントによって規定される。そのフットプリントが出力画像中に既に表示されているフットプリントと重ならない場合には、ブラッシュストロークはその曲線に沿うようにのみ配置される。ブラッシュストロークが配置されていない曲線は廃棄され、ステップ3607と3608のプロセスが、曲線がなくなるまで(若干の修正を加えられながら)繰り返される。ステップ3607における若干の修正は、図270の曲線C1から曲線をオフセットさせる場合に必要な、半分のブラッシュストロークよりも大きく曲線をオフセットさせることである。

【1698】

上述の方法は、画像中の興味のあるオブジェクトから一連のブラッシュストロークが放射されるようにすることにも利用できる。

【1699】

与えられたサイズのブラッシュストロークで画像が覆われるのに続いて、より小さなブラッシュストロークでさらなる処理がなされ、画像においてブラッシュストロークの重要な特徴をさらに正しくするように、閾値レベルが高められる。そのようなテクニックは、細かさが要求される特定の部分にゴッホが使用するテクニックに近似している。

技術26の説明

【1700】

何らかの画像(例えば、写真)のために、該写真等の中の興味を引く領域(例えば、顔等の位置)を割り出せることが望まれる。一旦、画像中における顔の位置が特定されると、図271に示されるように、顔領域に対して吹き出しを配置するような、多くの奇抜な効果を実行する画像処理を単純にできる。顔領域が一旦検出されると、眼鏡のような顔に関連する物を配置する等、顔を変形させるというような他の効果を履行することができる。

【1701】

前述したように、画像における領域の検出は、領域に限定して芸術的效果を適用するというような、領域特異的処理に適用できる。例えば、背景の空の領域は幅広のブラッシュストロークで描き、最前面の顔の領域は細かいブラッシュストロークで描くと言った処理である。また、画像における特徴の決定等である。色を基準にした領域の検出は、検出される色が肌の自然な色合いであって、画像における人の顔の検出を行う時に極めて役に立つ。

10

20

30

40

50

【1702】

照明を変化させると、特定のスペクトルの色のオブジェクトが、輝度や彩度が変化されて画像中に現われるかも知れない。色相はオブジェクトの優位周波数（卓越周波数、dominant frequency）を評価するので、オブジェクトの色は、色相よりも輝度や彩度においての方が大きく異なる傾向がある。図272に示されるように、さらなる改善の方法3801は、輝度や彩度から色相を区別する色空間に、取得した画像をまず変換するという第1のステップを有する。そのようなもので、良く知られている2つの色空間としてはHLS色空間とHSV色空間がある。これらの色空間を含む様々な色空間は、フォーリー、J. D.、ヴァン・ダム、A.、フェイナー、S. K.、ヒューズ、J. F.、コンピュータグラフィックス、原理と実行、第二版、アディソンウェスレー、1990年、584-599ページのような標準的なテキストで十分に説明されている。

10

【1703】

次のステップは、多数の入力カラーシード（color seed）を入力することである（3804）。各シード3805は次のように定義される。：

- (1) 色値のシードカラー範囲
- (2) 局所的な色の差の制限（a local color difference limit）
- (3) 全体的な差の制限（a global color difference limit）

【1704】

次のステップ3808はビットマップを構築するためにシードを処理することである。

20

【1705】

図271の方法における最後のステップ3807は、入力画像と同じ容量であって、入力画像中の画素が、シードのリスト3805により定義される色域に属することを示すビットマップを出力することである。この方法3801は、別々のパス3808の各シードの処理と、各パスの出力ビットマップと全体の出力ビットマップの結合を含む。個々のビットマップのどんなセット・ビットも全体のビットマップに書かれている。

【1706】

図273を参照すると、図272のステップ3808がより詳細に示されている。第1のステップ3820では、問題になっているシードが定義されるシードカラーレンジにシードがある全ての色について、シードビットマップが割り出される。次のステップ3821では、該シードビットマップにおける各シードのため、シード位置に連結される入力画像中の4コネクタ又は8コネクタの領域を検出する“シード充填”アルゴリズムのためのシード位置として利用される。そのシード充填アルゴリズムは、シード出力ビットマップの領域に属する画素にフラッグを立てる。ステップ3821にてシード出力ビットマップ中でフラッグが立てられたシードは、それらが次のシード充填のためのスタートポイントとして使用されないように、ステップ3820で作成されたシードビットマップから消去される。

30

【1707】

全てのシードがシード充填処理のために使用されると、シードの全てのリストのために、シード出力ビットマップは全体的な出力ビットマップと結合される。

40

【1708】

上述の参考文献の979頁～986頁に開示されているように、シード充填アルゴリズムは標準タイプである。しかしながら、該シード充填アルゴリズムは特定の画素関係テストを有する。

【1709】

もし、色が、前述の画素の色と局所的色差制限と同程度までに異なり、シードカラーと全体的色差制限と同程度までに異なるならば、画素は現在の領域の一部となる。その色差制限は各色成分毎に別々に決められる。スパンは検出されるか、左右から処理されるので、前述の画素は現在の画素の左の画素である。新しいスパンの初めの部分で、前述の画素

50

の色は、もとのスパンの平均色となるようにされる。シードカラーは、シードカラー範囲の中間色となるようにされる。

【1710】

区別できる色の特徴であるが良く知られた顔のような領域を同定するために、1つのシードを使用するのではなく、シードのリストが使用される。

【1711】

局部的色差制限及び全体的色差制限の使用は、画像中の不連続を横切らずに、可能な限り領域を検出するために使用される。領域は様々な色を含むかも知れないが、これらは、その領域にわたって穏やかに変化する。領域における全体の色の変化は、局部的な色の変化よりも一般的に大きい。局部的色差制限の使用(imposing)は、画像の不連続部を横切るというリスクを減少させる。また、それは、全体的色差制限が控えめにならないことを許容する。その領域の検出は、例えば、徐々に明るく或いは暗くなる領域においてでさえより完全にできる。

【1712】

入力画像の色空間が色相を基準としている場合、低輝度又は低飽和のため、色相は少々当てにならないので、色相差制限は好ましくは輝度(又は輝度値)及び飽和によって縮小される。したがって、領域中の暗いエリアや飽和していないエリアにおいて色相が当てにならない場合には、検出された領域にそれらのエリアが含まれるようにその効果は縮小される。これは、もちろん、輝度及び飽和に関しての差制限が前提となっている。

【1713】

また、色相六角形の円の特性に一致して、色相差は正確に計算される。

【1714】

画像中のオブジェクトを検出する上記アルゴリズムを使用した場合に、非常に適切な結果がしばしば得られることは分っている。例えば、上記の方法は、多くの固体を有する画像であって、カラーシードが標準的な皮膚の色調にセットされた画像に利用された。その結果は、画像中の個々の顔の位置を正確に探すことができる。

技術27の説明

【1715】

さらなる改善では、図274に示すように、ソースとなる画像3901が、対応する“タイル状”のカラー画像3902に変換される。その変換処理については後述する。第1の実施例では、各タイル3904は、出力画像3902において標準的配列を取る。さらに、各タイル3904の色は、入力画像3901における対応画素3905の色から引き出したものである。出力画像におけるタイル化処理は、入力画像3901よりも高解像度で行われ、よって、そのタイル化処理は実際において低解像度画像にするために使用される。

【1716】

もちろん、タイルの形状としては、多くの異なるものが可能である。画像をタイル化するタイルの形状は、好ましくは、全体の画像をタイル化して複製する場合に“パターンユニット”に結合されると良い。例えば、図276に示すように、タイル3908と3909は、複製される際に画像をタイル状にして、パターンユニット3907として結合される。

【1717】

さらなる実施例においては、一つ一つのタイルは2つのチャンネルを持つ画像により規定される。第1のチャンネル3970は、図276に示すように、タイルの形状と不透明度を規定する。零でない不透明な領域3971は、タイルの一部である画素を規定する。各画素3971の値はその画素の不透明度を示す。図277及び図278を参照すると、第2のチャンネル3980は、高さ(height field)として、タイルにおける画素表面テクスチャを定義する。図278では、図277のA-A'線に沿った断面図が示されており、強度やピーク3991や傾斜3992, 3993が示されている。

【1718】

そのアルゴリズムへの入力は以下の通りである。すなわち、
タイルとして再生されるべきカラー画像
上述したようなタイルのパターン
出力解像度

【1719】

付加される光学上の入力パラメータは下記のものである。

【1720】

該アルゴリズムからの出力は、入力画像にタイル状の演出を行うことである。

【1721】

繰り返しのタイル“パターンユニット”は次のように規定できる：

10

- (1) パターンユニットのサイズ（行の数、列の数）
- (2) 画像の第1画素のため、パターンユニットにおけるスタートポイントへのオフセット（行及び列のオフセット）
- (3) 連続するパターンユニットの配置についての、垂直方向のオフセット
- (4) パターンユニットを構成するタイルのリスト
- (5) 1つのパターンユニットにおける、水平方向の連続的なタイルの配置のオフセット
- (6) 該パターンの解像度（例えば、該パターンのタイル画像の解像度）

【1722】

該パターンの幅は、水平方向に連続的に配置されるタイルの列オフセットの合計により定義される。

20

【1723】

全てのオフセットは、連続する画素の組み合わせ（例えば、実数値の行及び列のオフセット）の中にある。このことは、タイルがサブ画素の正確さで配置されることを許容する。このことは、入力画像や出力画像や独立するタイルパターンの解像度を許容する。

【1724】

基本的なタイリング・アルゴリズムは、次のC++コードフラグメントにより実施される。すなわち、：

Tile

```
(  
    Image& inputImage,  
    TilePattern& pattern,  
    double outputResolution,  
    Image& outputImage  
)  
{  
    // initialise output image  
    double inputToOutputScale =  
        outputResolution / inputImage.Resolution();  
    int outputHeight = (int)ceil(inputImage.Height() * inputToOutputScale);  
    int outputWidth = (int)ceil(inputImage.Width() * inputToOutputScale);  
    outputImage.Initialise(outputHeight, outputWidth, outputResolution);  
  
    // scale pattern to input resolution  
    double patternToInputScale =  
        inputImage.Resolution() / pattern.Resolution();  
    inputPattern = pattern;  
    inputPattern.Scale(patternToInputScale);  
  
    // scale pattern to output resolution  
    double patternToOutputScale =  
        outputImage.Resolution() / pattern.Resolution();  
    outputPattern = pattern;  
    outputPattern.Scale(outputScale);  
}
```

10

20

30

40


```

// compute output to input scale factor
double outputToInputScale =
    inputImage.Resolution() / outputImage.Resolution();

// tile image
double patternStartRow = -outputPattern.StartRow();
double patternStartCol = -outputPattern.StartCol();
for
(
    double row = patternStartRow;
    row < (double)outputImage.Height();
    row += outputPattern.RowOffset()
)
{
    double tileStartRow;
    for
    (
        double col = patternStartCol,
        int nextTile = 0,
        Tile inputTile = inputPattern.Tile(nextTile),
        Tile outputTile = outputPattern.Tile(nextTile);
        col < (double)outputImage.Width();
        col += tile.ColOffset(),
        nextTile = (nextTile + 1) % pattern.TileCount(),
        inputTile = inputPattern.Tile(nextTile),
        outputTile = outputPattern.Tile(nextTile)
    )
    {
        // reset tile start row at start of pattern
        if (nextTile == 0)
            tileStartRow = 0;

        // compute tile position in input
        double tileInputRow = row + tileStartRow;
        tileInputRow *= outputToInputScale;
        double tileInputCol = col;
        tileInputCol *= outputToInputScale;

        // compute integer and fractional position

```

```

int intTileInputRow = (int)(tileInputRow + 0.5);
double fracTileInputRow = tileInputRow - intTileInputRow;
int intTileInputCol = (int)(tileInputCol + 0.5);
double fracTileInputCol = tileInputCol - intTileInputCol;

```

```

// scale input tile image
Image inputTileImage = inputTile.Image();
Translate(inputTileImage, fracInputTileRow, fracInputTileCol);

```

10

```

... determine the color of the tile from the input image
... i.e. inputTileImage at (intTileInputRow, intTileInputCol)

```

```

// compute tile position in output
double tileOutputRow = row + tileStartRow;
double tileOutputCol = col;

```

```

// compute integer and fractional position
int intTileOutputRow = (int)(tileOutputRow + 0.5);
double fracTileOutputRow = tileOutputRow - intTileOutputRow;
int intTileOutputCol = (int)(tileOutputCol + 0.5);
double fracTileOutputCol = tileOutputCol - intTileOutputCol;

```

20

```

// scale output tile image
Image outputTileImage = outputTile.Image();
Translate(outputTileImage, fracOutputTileRow, fracOutputTileCol);

```

```

... composite the tile with the output image
... i.e. outputTileImage at (intTileOutputRow, intTileOutputCol)

```

30

```

// update tile start row
tileStartRow += outputTile.RowOffset();

```

```

}

```

```

// update pattern start col
patternStartCol += outputPattern.ColOffset();
while (patternStartCol > -outputPattern.StartCol())
    patternStartCol -= outputPattern.Width();

```

40

```

}

```

```

}

```

【 1 7 2 5 】

上述の通り、一つ一つのタイルは、その形状／不透明性チャンネルにより特徴付けられる形態を有している。そのタイルの形状並びにタイルの画像中における配置は、一緒になって、そのタイルによって覆われるところの、入力画像中の一連の画素を決定する。出力画像における、そのタイルの色は、その一連の画素に基づく。色を付与するという機能は、限定するものではないが、入力される色を1画素ずつコピーしたり、入力される色の平

50

均を取ったり、入力される色の中央を取ったりすること等を含むようにしても良い。

【1726】

このように色が付されたタイルは、不透明度を無視して出力画像に直接描かれるかも知れないし、或いは、タイルの不透明度に従って出力画像に合成されるかも知れない。後者のケースでは、出力画像は最初に何らかの明度（例えば、黒）や何らかの任意の画像（例えば、入力画像自体）に初期化されなければならない。不透明なマスクやチャンネルを使用した合成画像は、例えばTomas Porter and Tom Duff（1984）“合成デジタル画像”SIGGRAPH会報1984年、253ページから259ページのような一般的な記事に開示されている標準的な合成技術を使用することができる。

10

【1727】

それぞれのタイルは、テクスチャチャンネルにより定義される表面テクスチャマップを有している。そのテクスチャチャンネルは、タイル中における各画素の相対的な表面高さを規定する。この高さ（height field）は、模擬的な入射光の反射角決定に使用される表面法線を決定するために後で使用される。タイルや、そのタイルが配置される部分の画像に模擬的な光が与えられる場合、表面テクスチャは視覚的により明白になる。

【1728】

タイルが出力画像に直接描かれるか、出力画像に合成されるか、いずれにせよ、最も単純なケースでは、タイルのテクスチャは出力画像のテクスチャチャンネルに直接書かれる。

20

【1729】

各タイルの不透明チャンネルは、タイルの背景への合成を制御する。加えて、画像サイズの不透明チャンネルは、タイル全体の背景への合成を制御するかも知れない。最も単純なケースでは、タイルの不透明度は、タイルの合成に使用される不透明度を生じさせる、大域的な不透明度（global opacity）によって測られる。図279を参照すると、大域的な不透明度を使用するプロセスの一例が示されている。第1のプロット4000では、折り返された不透明度プロフィール（例えば、4001、4002等）の単純な形状を持つ各タイルの、断面の輪郭の例が示されている。次に、不透明度4005の段階的增加からなる、好ましい大域的な不透明度チャンネル4004が示されている。上述した単純なケースでは、増加している不透明度のエリア（例えば、4007）を有する不透明度4006を生じさせるため、大域的な不透明プロフィール4004でタイル不透明プロフィール4001が測られる。

30

【1730】

もちろん、大域不透明チャンネルを伴った局所タイル不透明度の組み合わせのスキームは可能である。例えば、最初に、各タイルについて、画素の不透明度が平均化され、画素毎の不透明度は、画素毎の大域的な不透明度ではなく、この平均化された不透明度によって測られる。

【1731】

上述したタイリング・アルゴリズムが、僅かな簡単な修正を行うことによって、描いたような効果を簡単に生じさせる上で役立つということは、驚きを持って発見されている。パターンユニットをブラッシュストロークの堆積として取扱い、各タイル画像がブラッシュストローク（例えば、ブラッシュヘアーの層や、ストローク端部で厚みを有するもの）に見えるようにすることにより、ストロークがオーバーラップされて、描いたような効果がタイル内やパターンユニット内に実行される。次に、このアルゴリズムは、連続的にパターン内でタイルを反復するよりもむしろ、無作為にストロークコレクションからストロークを選択するために修正される。さらに、下に描かれた、多くのテクスチャリング、配置、彩色効果は代替的付加的に供給され得る。

40

【1732】

第一に、入力イメージ内に一般的には様々なレベルの細部がある。入力イメージは、タイリング又は彩色アルゴリズムを介して再現されたとき異なったレベルの細部を要求する

50

。これは、多様な変化内でイメージの加工、それぞれの変化内の異なった程度のストロークコレクションの利用、それぞれの変化と（累積的な）バックグラウンドの合成によって達成され得る。画素単位の詳細な寸法で構成された、イメージサイズにされた細部マップは、それぞれの変化過程のイメージ部分を制御するために利用され得る。それぞれの変化は、その上でイメージの対応部分が処理される“細部境界”が与えられる。継続的な変化に、細部のレベルの増加とストロークコレクション割合の減少が与えられる。最初の変化はこのように、例えば全体のイメージを加工処理し、第2の変化は細部の確定したレベルを加え、第3の変化はなお多くの細部を加える、などである。これは、画家がどんどん細部を加えていくことによって絵画を精密にしていくのにいくらか類似している。この細部マップ自体は、イメージ内にエッジを配置すること、言い換えると派生したフィルタの間をイメージが通過することによって、簡単かつ自動的に生成され得る。

10

【1733】

タイルは概念上は柔軟性の無いもの（rigid）である。その表面のテクスチャは、背景の表面テクスチャや、重なり合う他のタイル（通常は如何なる他のタイルとも重なり合わないが）の表面テクスチャからの影響を受けることはない。他方では、ブラッシュストロークは概念的には柔軟である。その厚さや表面テクスチャは、背景の表面テクスチャや、重なり合う他のブラッシュストロークの表面ストラクチャの影響を受ける。特に、そのブラッシュストロークは、背景の凹部に充填される傾向があるが、背景の頂点（peak）では薄くなる傾向がある。（背景は、そのストロークに先行するブラッシュストロークの影響を含んでいる。）

20

【1734】

別の実施例においては、ブラッシュストロークが描かれる場合、幾つかの方法の内の一つによって、そのテクスチャは背景のテクスチャと結合される。まず、新しい背景高さを生じさせるために、ストローク高さは背景高さの一定割合（例えば、25%）だけ付加されるが、その高さは減少しないようにされる。このテクスチャの効果は図280(a)から(c)を参照して説明される。図280(a)においては、シングルタイルのブラッシュの断面輪郭の例が示されている。図280(b)においては、図280(a)のブラッシュパターン4010を覆う、画像部分の断面が一例として示されている。図280(c)においては、上述したルールを使用した結合処理の結果が示されている。線分4013, 4014, 4016, 4017は、画像値4011が対応するブラッシュ値4010を超えるように、画像4011から直接取られたものである。しかしながら、4015の部分は、画像4011の対応部分の25%に、ストローク4010のストローク高さを加えて計算されたものである。ストロークの下背景が計算される。個々の画素において、背景の高さがもし平均よりも低ければ、ストローク高さは単に背景高さに加えられる。このようにして、背景の凹部にストロークが“充填”されるような擬態が行われる。もし、背景高さが平均に等しいか、平均よりも高ければ、ストローク高さはその平均に加えられる。このようにして、背景のピークによってストロークは薄くなる。背景がストロークを実質的に打ち破らない（例えば、ゼロまで薄くならない）ようにするため、その高さは、最小量（もちろん、ゼロを含む、最小ストローク厚さ）に応じて増加するようにされている。

30

【1735】

さらに、テクスチャー結合アルゴリズムの結果である、ストロークの正味の薄さを計算するため、ストロークの不透明度が計られるかも知れない。このようにして、背景のピークによりストロークが薄くなる場所は、より透明になる。

40

【1736】

タイル／ストロークの位置が決定されると、さらに、多くの方法によって処理がされるかも知れない。例えば、手によるタイルやストロークのポジショニングにおける自然なバリエーションを真似るため、タイル／ストロークの位置は、特定の範囲や特定の速度で任意にジッターされるかも知れない。ストロークよりもタイルを真似た場合、通常は、タイルが重ならないように、タイルのジッターの範囲が限定される。例えば、“内部タイルのギャップ”の幅のみが変化される。ブラッシュストロークを真似る場合、ブラッシュスト

50

ロークは通常は重なるので、ストロークジッター範囲は通常はより狭く限定される。

【1737】

さらに、タイルやストロークの位置についてのより標準的なバリエーションを真似るため、置換マップに従って各タイルの位置が置換される。その置換マップは2つのチャンネルからなる。第1は行の置換を含み、第2は列の置換を含む。

【1738】

タイル／ストロークの色自体は、芸術的な効果のためにさらに処理されるかも知れない。例えば、

- (1) 制限されたタイルや制限されたペイントパレットを真似るため、その色は、任意のパレットにマッピングされるかも知れない。
- (2) タイルカラーの不完全性や、絵の具を混合するバリエーションを真似るため、その色は、特定の範囲及び特定の速度で任意にジッターされるかも知れない。
- (3) その色のクロミナンスは、特定の速度で任意に変換されて、反対色を使用した印象派のスタイルを真似るかも知れない。

【1739】

タイル／ストロークの平均大域不透明度が決定されると、芸術的効果のためにさらに処理されるかも知れない。例えば、平均化された不透明度が、ゼロの不透明度や100%の不透明度に二値化されるかも知れない。このことは、タイルの不存在や存在を作り出す。もし、平均化された不透明度が二値化されたなら、大域的不透明度の再生は、入力画像に部分的に誤差拡散を行うことによって改良されるかも知れない。これは、画像中に重要な効果を生成するために見出されている。

【1740】

さらなる実施例が、アートカム装置の適切なプログラミングにより実行される。キャンバスに画像を“描く”ときの画家は、通常は、絵の具の色が限定されたパレットと、制約されたセットのものであって特徴的な筆遣いを形成するブラシとを用いて仕事をし、そして、独特の色や質感のキャンバスに描いている。そのことに気付くことにより、さらなる実施例が構築される。描くシーンをより真実に近付けた真実から遠ざけたりするため、そのシーンについての画家のビジョンを描くため、画家は、異なる色の筆遣いを行う。いくつかの絵のスタイルが、シーンの特徴や色を再生するために試みられる。他のスタイルが、シーンの特徴や色についてのより多くの印象を取得するために試みられ、より単純で一様なブラッシュストロークや絵の具を混合したり重ね塗りしないことによって特徴づけられる。これらの印象派のスタイルは、写実派のスタイルに比べるとより明るくより鮮明な色を達成するが、詳細な描写は犠牲にされる。

【1741】

“描かれた”演出となるように写真画像を自動的に変換するコンピュータシステムが、一般的には、入力画像中の画素から何らかの方法で取得した色のブラッシュストロークを出力画像中に入れる。そのようなシステムの課題は、上述したような、人が描くという行為の態様を取り入れることにある。印象派の演出を作り出す部分的解決法は、実際の芸術家の絵の具の色に基づくカラーパレットの構築と、入力画像に可能な限り一致させるような該パレットからのブラッシュストローク色の選択と、入力画像と整合する全体的な色を維持するための色の誤差拡散とにある。

【1742】

さらなる実施例においては、生の色だけを基準とするのではなく、ブラッシュストロークの最終的な色を基準とした画像を誤差拡散させるアルゴリズムが構築されている。この最終的な色は、絵の具の色や、ブラッシュの構成や、キャンバスの色や生地によって決定される。

【1743】

該アルゴリズムの入力は以下の通りである。すなわち、

- * 絵の具の色が限定されたパレット
- * それぞれが不透明マップ (opacity

10

20

30

40

50

map) やバンプマップ (bump

map) にて定義された一連のブラッシュストローク。バンプマップの使用は三次元グラフィック演出の当業者に良く知られていて、例えば、Graphic Gems の 4 巻 4 3 3-4 3 7 ページに記載されている。

＊ 色画像やバンプマップにより規定されるキャンバス

＊ 絵画として再生されるカラー画像

【1744】

前記アルゴリズムの出力は、入力された画像の演出、つまり、ペイントパレットによって着色されたブラッシュストロークでキャンバスに“描く”ことである。該キャンバスやブラッシュストロークはテクスチャーされていると共に、これらのテクスチャーはペインティング工程の間は結合されているので、最終的なペインティングは一方向的に照らされるかも知れない。

【1745】

描かれたような効果を成し遂げるため、出力画像と入力画像との間で、より優れた色整合を成し遂げるための基本タイリングアルゴリズム (tiling algorithm) が使用される。最初のプロセスは、ブラッシュストロークのパレットフィードバックアルゴリズムである。このプロセスは、ストロークのカラーパレットの表を作成して実行される。図 281 に示す表 4025 は、ブラッシュストローク B1, B2, B3 等により分けられている。ブラッシュストロークセット中の各ブラッシュストローク Bn のため、そして、ペイントパレットセット中の各ペイントカラー C1-CN のため、ストロークカラー S1-S4 は、図 282 のフローチャートを参照して説明される通り、次のように決定される。すなわち、

1. ステップ 4021 にて、対応するブラッシュストローク B1 を色付けるためにペイントカラー C1 が使用される。
2. そして、不透明マップとバンプマップとを使用して、色つきのブラッシュストロークが空のキャンバスに合成される。
3. ステップ 4023 にて、描かれたストロークの平均色が計算される。この平均色が、図 281 のストロークカラー S1 である。そして、ポインター (不図示) が、ストロークカラーパレット入力 S1 から、対応するペイントカラーパレット入力 C1 にポイントするためにセットされる。
4. そして、ストローク・カラー・テーブルが色によりソートされる。

【1746】

画像を描く場合、図 283 に示されるように、各ブラッシュストロークが通常の方法で選択される。そして、各ブラッシュストロークの所望の色が入力画像から通常の方法で決定される。しかしながら、ペイントカラーパレット中で最も近いペイントカラー Cn は使用されない。その替わり、そのブラッシュストロークのため、ストロークカラーパレットの中で最も近いストロークカラー Sn が使用される。次に、そのブラッシュストロークのためのペイントカラーとして、対応するペイントカラー Cn が絵の具の色として使用される。

【1747】

該ペイントパレット中で利用できる色の範囲は制限されているため、通常は、所望の色と選択できる色 Cn との間にはいくらかの違いがある。この違い、或いは誤差は、誤差拡散を行う通常の方法で、入力画像の近接した部分に誤差拡散されることが出来る。所望の色とペイントカラー Cn の違いの誤差は使用されない。代わりに、所望の色とストロークカラー Sn との違いである誤差が、この目的のため、ストロークを横断する平均色として使用される。実際のブラッシュストロークは他のストロークと重なるのに対して、ストロークパレット中のストロークは空のキャンバスに描かれるので、この平均色 Sn がストロークパレットにおける平均色と異なるかも知れないことに気付く。

【1748】

この方法は、他のペインティング・パラメータの効果が同じ方法で取り入れられるよう

に拡張することができる。これは、ペインティング・メディウムの種類（パイン油、チョーク、チャコール等）や、キャンバスの種類（テクスチャー紙、ガラス等）などを含む。各パラメータは、ストロークカラーパレットの多次元表に次元を加えるが、本質的なアルゴリズムは変化しない。

【1749】

芸術家は、時折、1つのブラッシュストロークに2色又はそれ以上を混合させる。印象派のスタイルでは、これらの色はしばしば十分には混合されず、また、混合されない色がストロークの分離された成分として視認される。

【1750】

上述したアルゴリズムにおいて、多数色のブラッシュストロークは1色のブラッシュストロークの一般化にすることができる。それは、多数の不透明度マップにより適宜されることができる。これらの不透明度マップは、ブラッシュストロークの共通の排他的領域のため、ゼロでない不透明度を定義する。それらは、全体的なブラッシュストロークの形状を規定する。そのアルゴリズムに次の修正を導入することにより、多数色のブラッシュストロークが提供される。

【1751】

一つ一つのブラッシュストロークのためにストロークカラーパレットが構築される場合、ペイントカラーパレットからのペンチカラーの可能な組み合わせでブラッシュストロークに色が塗られる。このことは、小さなペイントカラーパレットのため明らかに役立つ。各ストロークカラーパレットの入力は通常の方法で構築し、各ストロークカラーパレットは対応する多数のペイントカラーパレット入力気宇。画像を描く場合、通常の方法で、ストロークカラーパレット中で最も近いストロークカラーを見つける。しかし、ストロークは、多数のペイントカラーで色づけされる。

技術30の説明

【1752】

図284を参照すると、アートカム装置のプログラムのさらなる改善のステップが示されている。標準的な技法の通り、ブラッシュストロークが区分的ベジエ曲線として特徴付けられると想定される。ブラッシュストロークは所定の厚みを持つと想定される。そのブラッシュストロークは一連のベジエ曲線からなる。第1のステップ4211は各区分的ベジエ曲線を処理することである。それぞれのベジエ曲線は、標準的な技術を使用して、最初に折れ線に変換される（4212）。次に、“最も速い”エッジ（その意味は以下で明らかになる）が、所定のインクリメントにおける最も速いエッジを“出発する（step along）”ために使用される。その位置は、ブラッシュストローク4214を合成するために使用される。

【1753】

図285を参照すると、ブラッシュストローク4221の基礎を形成する第1のベジエ曲線4220が示されている。そのベジエ曲線4220を対応する一連の線分（例えば、4222）に変換するプロセス。折れ線の区分に線形近似するプロセスは標準的で良く知られており、普通の教科書に取り上げられている。その線形近似プロセスの一環として、各線の端点の法線（例えば、4224）がまた決定される。それらの法線は、必要ならば、線の傾きを補間する方法によって、点（例えば、4225）に隣接される線から決定されることができる。

【1754】

図286を参照すると、2本の線分4233, 4234の拡大図が示されている。各点（例えば、4232）のために、法線（例えば、4233, 4234）は、それぞれの側にブラッシュストロークの幅の距離だけ突き出ている。点4236は突き出た法線4237, 4238を有している。次に、法線4233と法線4237との間の距離4239が測定され、法線4234と法線4238との間の距離4240が測定される。エッジ4239に沿って移動するbodyが、エッジ4240に沿うよりも早く移動しなければならないという意味で、最も長い距離を有する側に“最も速い”エッジが定義される。したが

って、最も速いエッジは曲線の凸状のエッジである。

【1755】

最も速いエッジは、等しい量だけ離れたパラメータとしての所定数のポイントを定義するために使用される。図286の例は、線4239に沿って等間隔に規定された3つのポイント4243-4245が示されている。ほとんどの間隔は図286に示されたものよりも長い。

【1756】

次に、各ポイントのパラメータ位置が決定され、線4230に沿ったパラメータポイントを決定するために使用される。その位置は、ブラッシュストロークが配置される位置である。

10

【1757】

図287を参照すると、一連のブラシスタンプ4246, 4247, 4248が示されている。2つのスタンプ4246, 4248は線の端の部分に使用され、部分4247は線の中央部分に沿って連続的に使用される。

【1758】

該ブラシスタンプは、マットチャンネル、バンプマップチャンネル及びフットプリントチャンネルの3つの別々のチャンネルにより定義される。図287の例では、ブラッシュのマットチャンネルが示されている。“最大”テクニックとして知られている、第1のテクニックにおいては、マットチャンネルやバンプマップチャンネルは、分離されたブラッシングバッファにおいてブラッシュストロークを構築するために使用される。この合成技術においては、ブラッシュは最大不透明度を取ることにより構築される。そのバッファにて新しい不透明度が古い不透明度を超える場合、その不透明度は新しい不透明度に置き換えられる。生成された結果は、フォトショップにおけるエアブラシやペインティングに近似したものとなる。

20

【1759】

第2の合成技術においては、“フットプリント”技術として知られる。現在のラインのための前述したブラッシュスタンプが画素のためのブラッシュバッファに書き込まれなかった場合にのみマット値が変更されるように、フットプリントチャンネルが使用される。第3の技術においては、マットチャンネルやバンプマップチャンネルが使用されるが、このとき、マットチャンネルやバンプマップチャンネルは最小である。

30

【1760】

上記の技術は、特に水彩画の効果を真似る場合に良い結果をもたらすことが分っている。次のコードセグメントは、さらに改良された実施例の処理を示す。：


```

/*
 * $RCSfile: G2PolylineNav2.h$
 *
 * G2_PolylineNavigator2 template class definition.
 *
 * Author: Paul Lapstun, 1996/11/7
 * Copyright Silverbrook Research 1996.
 *
 * $Log$
 */
10

#ifndef _G2PolylineNav2_h_
#define _G2PolylineNav2_h_

#include <bool.h>
#include <G2Polyline.h>
#include <G2VectorList.h>

template<class T>
20
class G2_PolylineNavigator2
{
public:
    G2_PolylineNavigator2(G2_Polyline<T> const&
    pl, G2_VectorList<T> const& tl, G2_Real halfWidth)
        : m_polyline(pl), m_tanList(tl),
    m_halfWidth(halfWidth) { Start(); }

    bool AtEnd() const { return m_pi == m_polyline.end(); }
    bool Empty() const { return m_polyline.empty(); }
    void Start();
30

```

```

void Advance(T s);
G2_Point<T> Point() const;
G2_Vector<T> Tangent() const;

private:
void SetEffectiveSegmentLength();

private:
G2_Polyline<T> const& m_polyline; // polyline being navigated
G2_VectorList<T> const& m_tanList; // tangent list being navigated
G2_Real m_halfWidth; // half 'brush' width
G2_Polyline<T>::const_iterator m_pi; // polyline iterator
G2_VectorList<T>::const_iterator m_ti; // tangent list iterator
G2_Point<T> m_p0; // point at start of segment
G2_Point<T> m_p1; // point at end of segment
G2_Vector<T> m_v0; // tangent at start of segment
G2_Vector<T> m_v1; // tangent at end of segment
G2_Vector<T> m_n0; // normal at start of segment
G2_Vector<T> m_n1; // normal at end of segment
T m_length; // length of current segment
T m_s; // displacement along current
segment
G2_Real m_t; // parameter along current segment
};

template<class T>
inline
void
G2_PolylineNavigator2<T>::Start()
{
// set iterators to beginning
m_pi = m_polyline.begin();
m_ti = m_tanList.begin();
if (AtEnd())
return; // empty...

// get segment start point and tangent

```

```

    m_p0 = *m_pi++;
    m_v0 = *m_ti++;
    m_n0 = m_v0.Normal().Normalise() * m_halfWidth;
    if (AtEnd())
        return; // shouldn't happen...

    // get segment end point and tangent
    m_p1 = *m_pi;
    m_v1 = *m_ti;
    m_n1 = m_v1.Normal().Normalise() * m_halfWidth;

    // set effective segment length
    SetEffectiveSegmentLength();

    // reset displacement and parameter at start
    m_s = 0;
    m_t = 0;
}

template<class T>
inline
void
G2_PolylineNavigator2<T>::Advance(T s)
{
    if (AtEnd())
        return;

    // skip segments until s falls in current segment
    while ((m_s + s) > m_length)
    {
        // get segment start point and tangent
        m_p0 = m_p1;
        m_v0 = m_v1;
        m_n0 = m_n1;

        // get segment end point and tangent
        m_pi++;
        m_ti++;
        if (AtEnd())

```

```

        return; // genuine end...
        m_p1 = *m_pi;
        m_v1 = *m_ti;
        m_n1 = m_v1.Normal().Normalise() * m_halfWidth;

        // allow for rest of previous segment
        s -= (m_length - m_s);

        // set effective segment length
        SetEffectiveSegmentLength();

        // reset displacement at start of segment
        m_s = 0;
    }

    // move along current segment by remaining s
    // and compute parameter at new displacement
    m_s += s;
    m_t = m_s / m_length;
}

template<class T>
inline
G2_Point<T>
G2_PolylineNavigator2<T>::Point() const
{
    if (AtEnd())
        return G2_Point<T>(0, 0);
    return m_p0.Interpolate(m_p1, m_t);
}

template<class T>
inline
G2_Vector<T>
G2_PolylineNavigator2<T>::Tangent() const
{
    if (AtEnd())
        return G2_Vector<T>(0, 0);
    G2_Vector<T> tangent = m_v0.Interpolate(m_v1, m_t);

```

```

    return tangent;
}

template<class T>
inline
void
G2_PolylineNavigator2<T>::SetEffectiveSegmentLength()
{
    // the normals at the end points of the current line
    // segment have already been computed and scaled in
    // length by half the 'brush' width; now construct
    // vectors which join the ends of the normals; these
    // vectors are parallel to the line segment which
    // approximates the curve; the vector on the convex
    // side of the curve is longer than the line segment;
    // the vector on the concave side is shorter; choose
    // the length of the longer vector for displacement
    // stepping; note that a parameter t computed at a
    // particular displacement along the chosen vector
    // can be used directly to compute the point on the
    // line segment which approximates the curve
    G2_Vector<T> vectorA = (m_p1 + m_n1) - (m_p0 + m_n0);
    G2_Vector<T> vectorB = (m_p1 - m_n1) - (m_p0 - m_n0);
    G2_Real len2A = vectorA.LengthSquared();
    G2_Real len2B = vectorB.LengthSquared();
    if (len2A > len2B)
        m_length = sqrt(len2A);
    else
        m_length = sqrt(len2B);
}

#endif /* _G2PolylineNav2_h_ */

```

図288を参照すると、アートカム装置4302を含む、さらなる改善の配列(4310)が示されており、該アートカム装置4302は、適切な文字認識のできるタッチパッドLCDからなる文章入力装置4303に接続されている。代わりに、キーボード入力装置(例えば、4304)を文章入力装置とすることができる。文章入力装置の好適な形態は、前記アートカム装置4302に接続されるように適応されプログラムされたアップルのニュートン(登録商標)装置にて構成され得る。また、文章入力装置4303の他の形態が利用されることもできる。さらに、アートカム装置4302に挿入されるアートカード装置4305が提供され、該アートカードの表面に描かれている図に従って検知画像を(予め十分に検討されたように)処理するようになっている。

【1762】

図289を参照すると、さらなる改良版の操作に関しての好ましい形態が示されている。この操作の形態においては、アートカード4305は、ローマ字で規定されるフォントと、該フォントで特別の文字を作成する方法の説明とを含むVarkスクリプトで符号化されている。例えば、その説明は、そのフォントで新しい文字を作成するために外形を処理する方法を含む。

【1763】

入力装置4303、4304は入力装置フォントを有する。その入力装置フォントは、情報を表示するためにテキスト入力装置4303、4304によって使用される。特に、非ローマ字が使用される。したがって、入力装置4303、4304は、アートカード4305によって要求されるテキストの入力のために使用される。入力に基づき、フォントのアウトラインが、特別の文字の作成のためにアートカード4305に符号化された指示に従ってアウトラインを処理するアートカムユニット4302にダウンロードされる。それらの文字はアートカム装置4302によって作成され、印刷される出力画像の一部として表示される。

【1764】

この操作方法を使用することで、アートカム装置4302やアートカード装置4305にフォントの配列を保存することになしにアートカム装置4302の柔軟性が拡張される。この方法によれば、非ローマ字言語形式でアートカム装置4302を操作するために利用されるモデルの複雑さを減少すべく、テキスト入力装置(例えば、4303、4304)が各国ごとであることのみが必要である。

技術33の説明

【1765】

さらなる改善では、デジタル画像カメラで撮影し、前述の特許出願にて開示されているような所定のdpi(例えば、1600dpi)を持つインクジェットプリンターのようなプリント装置で印刷するという、カメラ画像の再生が望まれることが前提である。

【1766】

図290を参照すると、そのようなデジタル画像カメラ装置4501が、標準的な出力解像度の画像4502をプリントアウトするように設計されている。その画像4502は、マルチカラー出力(シアン、マゼンタ、イエロー)であって、入力画像の各色成分のためのドットの配列からなるインクジェットプリンター装置によって印刷される。出力された写真4502のコピーを希望する場合には、まず画像がスキャンされ、適切な色成分が抽出される。

【1767】

さらに改良された方法が図291に示されている。この方法では、写真画像4502がリニアCCD4510によって各色成分にスキャンされる。適切なリニアCCDは技術的に知られている。リニアCCD装置の構成及び操作の説明のために、「“CCDアレイ、カメラ、ディスプレイ” Gerald

C Holst 著、1996年発行、SPIE

オプティカルエンジニアリング出版」のような標準的な文献が参考とされる。さらに、適切なセンサー装置は、家電についての電気電子技術者協会報告書で定期的に述べられてい

10

20

30

40

50

る。

【1768】

CCDアレイ4510は、CCDヘッドの下を通過する画像をスキャンして信号を生成し、アナログ信号をデジタル信号に変換する。CCDアレイ4510は、写真のドット解像度の3倍である、1インチ当たり4800ドットにする。該CCDアレイ4510は、各色に対して一つずつ三原色、フィルター、CCDを有する。4800dpiは一連の千鳥配列CCDアレイを使用することにより達成される。

【1769】

前記データ値はフレームバッファコントローラ4512に転送され、色成分毎にフレームバッファ4513に保存される。保存された画像は、写真の上にプリントされ、前述したように、印刷時の約3倍の解像度であるCCDスキャン解像度にてCCDによりスキャンされるとき、本来のドットの位置に配置されるように処理される。残念ながら、スキャンされた画像には多くの欠陥が存在するかも知れない。これらの欠陥は、写真4502がCCDスキャナー4510に送られるときの写真4502の微々たる回転に加えて、写真4502の傷や歪みの影響を含む、スキャン工程に関連した欠陥を含む。

10

【1770】

パターンやドットを決めるための手順は、写真4502の回転を抽出するプロセスの使用に依存する。そのような回転を割り出す(determine)ため、多くの方法を利用することができる。

【1771】

例えば、インクドットのスキャンされたパターンは前記回転に基づく基本的特徴周波数(fundamental characteristic frequencies)を有する。写真には、ドットの一般的な配列でインクが印刷されるので、その画像のフーリエ変換が、起こり得る回転を割り出すために使用される。その代案としては、写真の境界(abrupt boundary)やCCDの下方のスキャン面(the underneath scanning surface of the CCD)からカードのエッジを割り出すことができる。さらに、画素の所望のドットピッチが決定(determine)される(1600dpi)。

20

【1772】

写真のスキャン画像の一端縁から、ドットがそれぞれの出力ドット位置に配置されているかどうかを判断するための処理がなされる。重要なことは、出力ドットの判定を行うように、カードを横切る同期を維持するという方法が使用される。1つのカラムから次のカラムへの間隙についての局所的な変化は極めて小さく、多くの変化は画像の長さを超えることは明らかである。

30

【1773】

回転や、隣接するドットの間隔を割り出す処理の1形態は、ドット中心(“重心”として知られたもの)の列を記録し(keep)、カラム中の重心をカラム方法によって書き換える(update)ことである。図292を参照すると、重心処理の一形態が示されており、画像領域4520は、カードの表面部分にて見本のインクドット4521を有している。各画素における各インクドットのサイズは、対応する画素サンプリングレートの約3倍である。一連の重心のマーク(例えば、4522)は、各カラムに保持(keep)されている。カラム C_{n+1} の重心マークにおいて初期の重心の回転を割り出すために、前に計算されていた重心マークや、隣接するカラム C_n の重心マークが利用される。例えば、重心マーク4523の初期位置を割り出すために、重心マーク4522, 4524, 4525が利用される。初期位置が割り出されると、さらなる調整が必要かどうかの判断のため、重心マーク4523の周りの画素値が調べられる。

40

【1774】

期待される位置から重心4523が動いているとの判定は、該ポイント4523の周りの画素値を調べることにより導き出される。その調査はX及びY方向で別々に行われ、その移動は、それらの方向で別々に行われる。

【1775】

50

多くの異なる方法が使用され得る。重心 4 5 2 3 の微小な修正が必要かどうかを判断するための 1 つの方法は、図 2 9 3 に従って述べられる。この方法によれば、限定された数のインクドット配列のみが可能であることに気付く。これらの可能な配列は、図 2 9 3 に 4 5 3 0 で示される通りのものであって、現在の画素 (current pixel) 4 5 3 1 と、隣接する 2 つの画素 4 5 3 2, 4 5 3 3 を示す。それぞれのインクドット 4 5 3 1 - 4 5 3 3 は、CCD センサーにて検知されるところの、ほぼ 3 つの画素値を有する。図 2 9 3 における各画素パターン 4 5 3 0 のため、デジタル変換された後の CCD 出力値 4 5 3 5 の例を示す。CCD のサンプリング効果や、他の機械的及び明るさの効果のため、ドットパターンの CCD 値 4 5 3 5 は、AD 変換されたガウス曲線 (或いは、その一部) と同等なものをしばしば含む。したがって、切り立ったエッジは通常は与えられない。しかしながら、期待された断面は、最も近い断面を割り出すために得られたものを調べることができ、誤差はそれほど大きくはなく、よりフィットするために制限内に調整される。この方法によれば、新しい重心位置は第 1 のディメンジョン内にて得られる。もちろん、重心を調整するという課題は 2 つの方向において対称的であり、同じ処理ステップが他のディメンジョンにおいても適用される。多数の他の技術が、見本画像からトレーニングされたニューラル・ネットワークのような、より発展した技術が使用されるかも知れない。

10

【1776】

調整した位置が割り出されると、重心 4 5 2 3 (図 2 9 2) が僅かに調整され、重心割り出しの処理が続行される。その新しい重心位置から、特定の重心位置のためにオン又はオフのインクドットの値が記録されるべきかどうかを判断するために周囲の見本の値が調

20

【1777】

図 2 9 1 に戻ると、記録されたインクドット値はインクドット配列 4 5 1 6 に保存され、同じ解像度 (1 6 0 0 d p i) か異なる解像度でプリンター (例えば、4 5 1 7 に示すものであって、オリジナル写真 4 5 0 2 に等しいインクドットの写真 4 5 1 8 をコピーするためのもの) にて印刷するために利用される。この方法によれば、ネガの使用や、画像をデジタルで分割して保存することをしなくても、オリジナル写真からのコピーである高品質写真のコピーが得られる。

【1778】

もちろん、処理のスピードアップと、大容量のデータのセットをフレームバッファ等に保存する必要を排除するため、カスタマイズされたリアルタイムのパイプラインの基本設計概念の利用を含む、多くの改善が可能かも知れない。

30

技術 3 4 の説明

【1779】

さらなる改善では、アートカム装置は、2 D モーションセンサーを含むように変更さる。そのモーションセンサーは、2 軸での動きを検出する小さな微小電気機械システム (MEMS) 装置や、他の好適な装置を備えることができる。そのモーションセンサーはカメラ装置に取り付けることができ、その出力はアートカム中央プロセッサによりモニターされる。

【1780】

図 2 9 4 を参照すると、さらなる改善の好ましい配列の概略図が示されている。加速度計 4 6 0 1 は、CCD 装置から不鮮明な検知画像を受け取るアートカム中央プロセッサ 4 6 0 2 に出力する。そのアートカム中央プロセッサ 4 6 0 2 は、写真が撮影されたときのカメラの角速度を決定するために、前記加速時計の測定値を使用する。そして、その測定値 (速度因子) は、好適にプログラムされたアートカードプロセッサ 4 6 0 2 によって利用され、不鮮明な検知画像 4 6 0 3 に対してボケ修正が行われ、ボケの無い出力画像 4 6 0 4 が出力される。アートカードプロセッサ 4 6 0 2 におけるボケ修正プログラムとしては、コンピュータプログラミングやデジタル画像修復の当業者により知られている標準的なアルゴリズムが使用される。例えば、"Selected Papers on Digital Image Restoration"、M. Ibrahim Seza

40

50

n編集、SPIE Milestoneシリーズ、74巻、特に再版の167-175ページが参照される。更に、簡易化された技術は“Image Processing Handbook”第2版、John C. Russ著、CRC出版発行、336-341ページで明らかにされている。

技術38の説明

【1781】

図295を参照すると、さらなる改善の方向に従った構成として、付属のプリンター5010の形態が示されている。該プリンター5010は、空洞5012の中に挿入されるプリントロール5011を有するように設計されている。該プリントロール5011は空洞5012に挿入されると共に、ローラー（例えば、5013, 5014）を有している。該ローラーは、内部の紙を挟むと共に、画像の印刷のため、要求に応じて該紙をプリントヘッド5015に搬送する。理想的には、該プリントロール5011は、消耗されるインクを供給する装置に加え、プリントメディアを備えている。インクは、画像の印刷の際にプリントヘッド5015により消耗される。

10

【1782】

プリンター5010は、アートカムの実施例の適切な再加工により構築されることができ。プリンターは、USBポートを介してコンピュータ5024と通信する。プリンターユニット5018はプラスチック成形のケースであり、画像を印刷するための内部プリントヘッド5015を有する。

【1783】

そのプリントヘッド5015は、自動ボンディング技術によりプリントヘッドチップ5020に接続されている。プリンターユニット5018のサイズを減少させるため、該プリントヘッドチップ5020はフレキシブルな回路基板に取り付けられている。USB通信ケーブル5022を介してプリンターユニット5018に転送される描写（description）から画像を生成するため、該プリントヘッドチップは、最新の半導体製造技術により作られ、関連したメモリー等を有している。もちろん、他の多くのインターフェースの形式が利用され得る。プリンターヘッドチップは、予め説明され得る。

20

【1784】

従って、画像の印刷が望まれるとき、プリンターユニット5018に接続されているコンピュータシステム5024は、ケーブル5022を介してプリントユニット5018にページ記述言語（page description）を転送する。プリントチップ5020は、後でプリンターヘッド5015で印刷される画像を準備する。プリントロール5011は、消耗されるプリントメディアやインクを、使いやすい形態で供給し、プリンターユニット5018により使用されると、そのプリントロールは新しいプリントロールに置き換えられる。

30

技術40の説明

【1785】

さらなる適応では、カメラシステム及び印刷システムは免除され、大きなスクリーンの読み取り装置（large screen reader）に置き換えられる。アートカードは、裏面に情報が書き込まれ、その情報の表示が他の面に印刷されるように設定されている。例えば、アートカードは本のコンテンツや新聞のコンテンツを含むことができる。そのようなシステムの一例が図296に示されるものであって、アートカード5210はその一面に本のタイトルを有し、他の面には、印刷されたコード化されたデータを有する。カード5210は読み取り装置5212に挿入される。この読み取り装置5212は、折り畳むことができるようにフレキシブルなディスプレイ5213を備える。カードの読み取り装置5212は、前や後ろにページをめくったり、他の制御を行ったりするためのディスプレイ制御装置5214を有する。

40

【1786】

したがって、図296の構成のものは、本や新聞や雑誌や技術マニュアル等の形式で情報を効率良く配布できることが分る。

【1787】

50

数多くの変更及び／又は修正が、大まかに説明された本発明の精神または範囲から逸脱することなく、特定の実施の形態に示されるような本発明へなされてもよいことは、当業者によって理解されよう。したがって、本実施例は、図示されていてもされていなくても、あらゆる点で考慮されるべきである。

技術 47 の説明

【1788】

さらなる改良は、擬似的なブラッシュストロークを使用してキャンバスの上に画像を“描く”という処理について論じられる。バンプマップ技術が使用されることが前提であり、特に、使用される各画像が、画像表面のテクスチャーを規定するバンプマップを有することが前提である。

10

【1789】

図 297 には、見本としてのラウンドブラッシュ・バンプマップ 5901 が示されている。図 298 では、図 297 の A-A' 線の断面が示され、該断面は、一般的なラウンドブラッシュストローク 5901 の表面の一般的外形 5903 を有する。図 297 のバンプマップを有するブラッシュストロークを一般的なヘシアン形の“キャンバス”画像（バンプマップは図 299 に示され、B-B' 線の高さは図 300 に示されたもの）に塗ることが前提である。その高さは、波状のピーク（例えば、5906）からなる。

【1790】

さらなる改善では、最終的なバンプマップの形成のために、図 298 のバンプマップと図 300 のバンプマップとが、供給された剛性因子に従って種々の方法で結合される。その剛性因子は、“剛性”のペイント、又は“フレキシブル”のペイントを使った効果を生じるように設計される。図 301 では、図 298 と図 300 の 2 つのバンプマップが低い剛性のブラッシュペイントのために結合された例が示されている。図 302 では、図 298 と図 300 の 2 つのバンプマップが高い剛性のブラッシュペイントのために結合された例が示されている。

20

【1791】

さらなる改善では、絵の具からなるブラッシュストロークが一定の可塑性を有していて背景の凹部に満たされるので、バンプマップの結合に際して、バンプマップは単純に付加されはしない。したがって、図 301 や図 302 の効果を成し遂げるためには、ブラッシュストロークのバンプマップは、背景のバンプマップであってローパスフィルターされたものに付加される。そのローパスフィルターは、絵の具の剛性因子によって決まる範囲（radius）を有している。該剛性因子が高ければ高いほど、フィルターの範囲は広くなり、透かして見える背景の表面テクスチャは少なくなる。ブラッシュストロークが薄かったり、何も無かったりする場所（ほとんどが縁部に沿っている）ではフィルターがかけられた背景の表面テクスチャが、実際の背景の表面テクスチャに収斂するように、ローパスフィルターの範囲はブラッシュストローク厚さに応じて変動される。

30

【1792】

図 303 及び図 304 を参照すると、初期の背景バンプマップを示す図 303 を用いた処理の例が示されている。バンプマップ 5911 の、ローパスフィルターがかけられたものを覆う状態の、剛性の高いペイント 5912 を図 304 は示し、ローパスフィルターがかけられたバンプマップ 5913 を覆う状態の、剛性の低いペイント 5914 を図 305 は示している。

40

技術 48 の説明

【1793】

さらなる改善では、入力画像の色域（color gamut）は出力画像の色域（color gamut）に“モーフ（morphed）”される。この出力画像の色域は、異なる複数の色域の試行という方法によって独断的に決定されたものである。さらなる改善では、理想的には、特定のスタイルのブラッシュストロークで画像を置き換えるというような、さらなる芸術的な処理にプレ処理ステップが使用される。そのブラッシュストロークは、画像から色を抽出し、該ブラッシュストロークは所望色域に関連する色を有する。

50

【1794】

最終的な出力画像を作成するステップは、図306に符号6010で示されたものである。第1のステップ6011は、望ましくはアートカム装置により検知された任意の入力画像を入力することである。その入力画像の色域は、所定範囲の色域の出力画像を作成するために“モーフ（Morphed）”又は“ワープ（Warped）”される（6012）。モーフィング処理については後述する。出力の色域が作成されると、次のステップでは、適正なブラッシュストローク技術が適用される。ブラッシュストローク技術は、大いなる変化をもたらすことのできる処理である。使用されるブラッシュストロークの実際の形態では、前述した技術が本発明に絶対必要である訳ではない。入力画像に対して、色域を制限した演出を行って出力画像を形成するためにそのブラッシュストローク技術が適用される。

10

【1795】

図307を参照すると、ガモットマッピングとモーフィングの問題の一例が示されている。L*a*b*色空間であるユニバーサル色空間6021で多くの可能な色値が定義されてシングルの色空間が使用されることが前提である。この色空間の内部で、入力センサーは、ある範囲の明度6022を検知することができる。さらに、出力プリント装置は、プリンター色域6023における特定の範囲の色を出力できるものと仮定する。該プリンター色域は入力センサーの色域6022よりも小さいかも知れないし大きいかも知れない。芸術的色域6024は特定のスタイルに従って規定される。その芸術的色域6024は、様々な技術の使用を通じて慎重に構築することができる。例えば、所望の出力画像がスキャンされ、所望の出力画像は、L*a*b*色空間の内部の特定の範囲を含むように、色のヒストグラムが構築される。所定の明度の3次元のL*a*b*色空間のように図307が解釈できることは、コンピュータグラフィック関係の当業者によれば理解され得る。

20

【1796】

したがって、入力センサー色域6022の中の色を芸術的色域6024に移す（map）ことが必要である。さらに、残念ながら、芸術的色域6024は、特定の出力プリンター色域6023にとって“はみ出る色域”の色を含んでいる。そのようなケースでは、出力の色がプリンター色域6023内にあるために、芸術的な色域6024を移す（map）必要があるだろう。

【1797】

図308を参照すると、主要な要求は、L*a*b*値6031をL*a*b*出力値6032に移すための3次元ルックアップテーブル（例えば、6030）をアートカードが有するということである。好ましくは、その3次元ルックアップテーブル6030は、3次元マッピングのために定義される特定のポイントと、定義されたポイント間の中間値のマッピングのために利用される3次補間とを有するコンパクトな形式で提供される。

30

【1798】

図309を参照すると、1つの実質的に任意の入力ガモットスペース6040から第2の所望の出力ガモットターゲットスペース6041へのガモット・モーフィングを適用するための処理についての極めて一般的な例が示されている。潜在的には、入力ガモットスペース6040の内側であって出力ガモットターゲットスペース6041の外側にあるポイント（例えば、6042）をガモットマッピング処理は処理しなければならない。前記3次元カラールックアップテーブル6030は $(2^n + 1)^3$ のサイズを有する。nは、1から、最大色コンポーネント・プレジジョン（例えば、通常は8ビット）までの範囲である。任意のワープがルックアップテーブル内で符号化される。該テーブルの使用は、また、別々に準備された符号化ワープファンクションを独立させるアルゴリズムを生じさせる。そのアルゴリズムは、任意のワープ関数のための予測できる性能を有している。ワープ関数が連続することは要求されない。したがって、そのアルゴリズムにおいては、任意のワープ関数にとって強固である。

40

【1799】

そのルックアップテーブルは、整数の3D配列であり、実数の出力色コーディネートは

50

入力コーディネートの順で配列される。このようにして、前方ワープ関数が符号化される。

【1800】

要求された画像のカラー・ワープが、1画素毎、出力画像の順に計算される。そのアルゴリズムは3Dカラーlookupテーブルへのランダムなアクセスを必要とするが、入力色は空間中では滑らかに変化するので、ランダムなアクセスは一般的には統一性 (coherent) がある。入力及び出力画像へのシーケンシャルなアクセスが要求される。

【1801】

不可欠なトリリニア・ワープ・アルゴリズムは次の疑似C++コードで実施される。

```

for (int row = 0; row < height; row++)
{
    for (int col = 0; col < width; col++)
    {
        outputImage[row][col] = Lookup(table, inputImage[row][col]);
    }
}

```

Colour

Lookup(LookupTable& table, Colour& color)

10

```

{
    // compute indices
    int fracPrecision = colorPrecision - log2(table.nSamples);
    int i = color[0] >> fracPrecision;
    int j = color[1] >> fracPrecision;
    int k = color[2] >> fracPrecision;

```

// compute interpolation factors

```

int one = (1 << fracPrecision);
int mask = one - 1;
double f = (double)(color[0] & mask) / one;
double g = (double)(color[1] & mask) / one;
double h = (double)(color[2] & mask) / one;

```

20

// trilinearly interpolate

```

Colour c000 = table[i+0][j+0][k+0];
Colour c001 = table[i+0][j+0][k+1];
Colour c00 = Interpolate(c000, c001, h);
Colour c010 = table[i+0][j+1][k+0];
Colour c011 = table[i+0][j+1][k+1];
Colour c01 = Interpolate(c010, c011, h);
Colour c0 = Interpolate(c00, c01, g);

```

30

```

Colour c100 = table[i+1][j+0][k+0];
Colour c101 = table[i+1][j+0][k+1];
Colour c10 = Interpolate(c100, c101, h);
Colour c110 = table[i+1][j+1][k+0];
Colour c111 = table[i+1][j+1][k+1];
Colour c11 = Interpolate(c110, c111, h);
Colour c1 = Interpolate(c10, c11, g);

```

40

```

return Interpolate(c0, c1, f);
}

```

【 1 8 0 2 】

前記ガモット圧縮プロセスは、ソースガモットにおける色の違いを維持しつつ、認知で

50

きる色の変化を最小限にするという方法で、ソースガモットの色を、より小さなターゲットガモットの色にマップさせようとする。ガモット圧縮の計算は色のワーピングから分断されるので、効果的なガモット圧縮は上述したルックアップテーブルで駆動されるプロセスを用いて実施される。

【 1 8 0 3 】

ガモット圧縮アルゴリズムは 3 次元ルックアップテーブルの構築を含み、図 3 0 9 を参照して述べたような次の疑似コードで実施される。

```
int fracPrecision = colorPrecision - log2(nSamples);
for (int i = 0; i < nSamples; i++)
{
    for (int j = 0; j < nSamples; j++)
    {
        for (int k = 0; k < nSamples; k++)
        {
            1. for constructing the lookup table at the color point at (i << fracPrecision, j << fracPrecision, k
               << fracPrecision);
            2. find nearest point 6043 on the luminance axis of the Lab space within the target gamut 6041
               of Fig. 309;
            3. construct a vector from the color point 6042 to a point on luminance axis 6043;
            4. compute the distance dp of the color point 6042 from the luminance axis 6043;(i.e. the length
               of the vector);
            5. compute the intersection 6045 of the vector with source gamut polyhedron 6040;
            6. compute the intersection 6046 of the vector with the target gamut polyhedron 6041;
            7. compute the distance dt from the luminance axis 6043 to the target gamut boundary 6046
               along the vector;
            8. compute the distance from the luminance axis 6043 to the source gamut boundary 6045 along
               the vector;
            9. compute the compression factor as ratio of target distance dt 6043 - 6046 to the source
               distance ds 6043 - 6045;
            10. compute the scale factor as the ratio of the color point distance dp 6043 - 6042 to source
                distance ds 6043 - 6045;
            11. scale the compression factor by the scale factor (so that in-gamut colors compress
                progressively less);
            12. scale the color point distance dp by the compression factor
            13. compute the point on the vector which is the compressed distance from the luminance axis;
            14. write this point to the lookup table at (i, j, k);
        }
    }
}
```

【 1 8 0 4 】

ソースガモット又はターゲットガモットのいずれかがパレットフォームにて知られているならば、ガモット多面体はパレットポイントの凸包から計算される。

【 1 8 0 5 】

図 3 1 0 を参照すると、芸術的効果の創造において一般的に活用される処理であって、後述の如くガモット・モーフィングと称される処理を使用して、何らかのソース・ガモット 6 0 5 0 をターゲット・ガモット 6 0 5 1 にマップさせるために上記処理が使用され得

る。ガモット・モーフィングは、ソース・ガモット中の色をターゲット・ガモット中の色へのマッピングを直接コントロールするために使用される。

【1806】

ガモット圧縮と同様に、ガモット・モーフィングは、画像の色域（ガモット）を特定の芸術的スタイルに似させるために使用されるかも知れない。ガモット・モーフの計算はカラーワーピング処理から分断されるので、ガモット圧縮と同様に、効果的なガモット・モーフィングは、ルックアップテーブルを駆動するアルゴリズムによって実施されることができる。

【1807】

ガモット・モーフィングと共に、多くのソース・ガモット色（例えば、6053）が、同数の対応するターゲットガモット色（例えば、6054）に直接マップされる。ルックアップテーブル中のマップされたターゲットの残存しているターゲット色は、特定のターゲット色（例えば、6054）の加重和として計算される。そのターゲット色は好ましくは、各ルックアップテーブルのポイントから対応するソースカラーまでの距離の二乗の逆数により重み付けされる。

【1808】

ガモット・モーフィング・アルゴリズムが次の疑似コードにより具現化される。

```
int fracPrecision = colorPrecision - log2(nSamples);
```

```
for (int i = 0; i < nSamples; i++)
```

```
{
```

```
    for (int j = 0; j < nSamples; j++)
```

```
    {
```

```
        for (int k = 0; k < nSamples; k++)
```

```
        {
```

```
            ... construct color point at (i << fracPrecision,
```

```
                j << fracPrecision, k << fracPrecision)
```

```
            ... initialise the weighted sum to zero
```

```
            ... initialise the sum of weights to zero
```

```
            for (int m = 0; m < sourceGamut.size(); m++)
```

```
            {
```

```
                compute the distance from point to a current source gamut point corresponding to m;
```

```
                compute the inverse of the distance squared;
```

```
                call the calculation from step 2 the weight, and add it to the sum of weights
```

```
                scale the corresponding target gamut point by the weight and add it to the weighted sum
```

```
            }
```

```
            ... divide the weighted sum by the sum of weights
```

```
            ... write this point to the lookup table at (i, j, k)
```

```
        }
```

```
    }
```

```
}
```

【1809】

前述の技術が最初に画像のガモットを所定のエリアに限定するために使用できることは、コンピュータグラフィックスにおける当業者には明白であろう。続いて、例えば、“点描画法”技術によって与えられると同様の効果を生じさせるため、制限されたガモット画像にブラッシュストロークフィルタが適用される。

技術53の説明

10

20

30

40

50

【1810】

前述のアートカムシステムの基本が図311に符号6501で示されている。そのアートカムシステム6501は、入力としてアートカード6503を必要とするアートカム6502に依存している。該アートカード6503は、出力写真6505の作成のため、画像のシーン（image scene）6504を処理するためのコード化された情報（指示情報）を有し、出力写真は、そのアートカード6503の指示情報に従って大幅に画像処理される。該アートカード6503は極度に高価とはならないようにされており、一面にはコード化された情報を含み、他の面には、アートカム6502に挿入された場合に生成されるであろう効果の描写を含んでいる。

【1811】

さらなる改善の方法に従って、図312に示すように、多数のアートカード6510が用意され、パックで販売される。各パックは、特定のサイズの衣服の着用に関連しており、カメラにより撮影された画像がマッピングされて衣服6512を着用したモデルの画像（例えば、6511）を有する。そのマッピングは、異なるアートカード6510で異なる衣服アイテムとすることも可能である。マッピングアルゴリズムの1つの形態が図313の6520に示される。該アルゴリズムでは、魅力的なワーピングの効果を生成する反復タイリングパターンに画像をマップするというワープマップを使用して、入力画像が最初にワープされる（6521）。もちろん、アートカード6503（図311）により与えられる、多くの他の形式のアルゴリズムが、材料の魅力的な形式を生成するために準備される。

【1812】

次に、第2のワープ6522は、第1のワープマップ6521の出力を、アートカード中の特定のモデルの画像にワープするために用意される。したがって、ワープ6522はアートカード特有である。その結果は、アートカム写真6505として印刷されるために出力される（6523）。したがって、ユーザは、望みの画像6504をアートカム6502に示すことができ、流行の衣服のモデルを有する画像の処理バージョンを有するアートカム写真6505を作成することができる。この処理は、希望の結果が得られるまで続けることができる。

【1813】

次に、布についての最終的な選択がなされた場合には、図314に示されるように、インクジェット布プリンタや関連の駆動制御電子機器等を有する布プリンター6534にUSBポートを介してアートカム6502が接続される（6503）。アートカム6502又はインクジェットプリンタ6534のどちらかが、布6535にガーメントピース（例えば、6536）を印刷するようにプログラムされている。ガーメントピースは、アートカードのモデルにより伝えられるのと同じ衣類を作成するため、オリジナルのワープ画像をその表面に有している。

【1814】

出力された布は、ガーメントピースを繋ぎ合わせるための指示6540に加えて、位置合わせと境界領域（例えば、6539）のためのタブ部（例えば、6538）を有している。好ましくは、その出力プログラムは、衣服の交差した継ぎ目上に連続して発生させるために縁取り部分のワープマッチングへの提供を含む。

【1815】

さらに、ユーザーインターフェースは、異なった形のボディーを考慮に入れるため、多くの異なった出力サイズを有する同じカードの利用に備えられる。アートカム技術の利用に基づいて、アートカム機構6502の利用によって、カスタムされた衣類の生産と適当な結果の迅速な描写のためにシステムを提供することが出来る。

技術54の説明***

技術56の説明

【1816】

さらなる改善では、画像の効果を転送するために、先述したようなアートカム装置が複

数、U S Bポートを介して接続されている。各アートカムの内部のコンピュータ部の適切なプログラムにより、画像効果の転送が成し遂げられる。

【1817】

好ましい配列は、図315に示されているように、一連のアートカム（例えば、6802, 6803, 6804）がそれらのU S Bポートを介して接続されているというものである（6805）。それぞれのアートカム6802, 6803, 6804には、好適な画像処理プログラムが格納されたアートカード6807, 6808, 6809が備えられている。さらに、ネットワーク環境を利用するという指示をアートカード6807, 6808, 6809により与えることができる。そして、アートカム6802により検知された画像6810は、アートカード6807上の処理プログラムによって処理される。アートカード6808によって画像処理機能が適用されるアートカム装置（一連のアートカム装置の内の次のアートカム装置）6803に、その結果が転送される（6805）。このような目的のため、各アートカム装置は2つのU S Bポートを持つように修正されている。最後のアートカム装置6804は、前述した画像処理の集積である出力6812を作成するため、アートカード6809に格納された処理を適用する。

10

【1818】

したがって、図315の構成6801は、1つの検知画像に対して多様な効果を与えることができる。もちろん、多くのさらなる改良は可能である。例えば、それぞれのアートカムは、次のアートカムへ画像を転送することに加えて、自身が処理した画像を印刷することもあり得る。加えて、経路を分離して、1つのアートカムを2つの異なった下流側アートカムに出力し、異なる最終画像を出力するようにする。加えて、ループ等も利用される。

20

技術57の説明

【1819】

さらなる改善では、開示されている技術は、双眼鏡セットに組み込まれるように適合されていて、双眼鏡のユーザからの要求により画像を印刷できるようになっている。まず、図316を参照すると、双眼鏡システムの一例が開示されている。該双眼鏡6901は、公知のタイプのものであり、本発明の原理に従って変形される。該双眼鏡は、レンズ入力を有する通常の光学レンズシステム6902, 6902を有する。2つのレンズシステムは、コネクタプレート6906にヒンジシステム6904, 6905を介して取り付けられている。その配列は大体従来のである。

30

【1820】

第1の光学レンズシステム6902はビーム・スプリッタ装置6908を有しており、該装置は、光学システム6902に入射された光の光路6909を2つの光路に分割する。第1の光路6910は、C C Dシステム6914が取り付けられた側に向けられる。第2の光路6911は、接眼部6915の方向へ向けられる。該ビーム・スプリッタ装置6908は、光線6911の出力強度を弱くさせることとなる。したがって、光学レンズシステム6902の中のビーム・スプリッタ装置6908による減衰と一致させるようにするため、減衰フィルター6918を配置しても良い。

【1821】

40

したがって、C C D撮影装置6914は、光学レンズシステム6902で見るのと同じシーンを撮影することとなる。該C C D装置6914は、アートカム中央プロセッサやメモリー装置を有するプリント基板6920に取り付けられた処理システムに接続されている。そのアートカム中央プロセッサ装置は、重要な画像処理機能を有しており、画像をキャプチャーするように前記C C D装置を制御し、プリントヘッド装置6923を使用して画像を印刷するようにする。プリンターには、プリントメディアにインクを吐出するインクジェットプリント装置を使用する。該プリントメディアは、プリントメディアと印刷用インクとを含む着脱可能な使い捨て型プリントロール6925から供給される“紙”フィルム（実際には、適切なポリマー）からなる。該プリントロール6925はハウジング6925に収容されており、該ハウジングにはバッテリー6927が着脱可能に配置されて

50

いる。

【1822】

ユーザがボタン6919を押し下げると、アートカム中央処理装置は、シーンを撮影するためにCCD6914を作動させる。CCD装置6914によってキャプチャーされた画像はACPプロセッサによって処理され、撮影したシーンを即座に永久的に記録すべく、プリントヘッドに転送されて即座に印刷される。プリントヘッドは再び上述のようになる。

【1823】

多くの改良が、さらなる改善を成し遂げることができる。例えば、アートカム中央プロセッサに接続されるUSBポートが提供されて、画像処理アルゴリズムを双眼鏡6901にプレロードできるようにしても良い。そのルーチンを作動させるための一連のボタンを配置するようにしても良い。そのルーチンは様々な画像強調操作等を含む。加えて、画像の特徴を強調するため、多くの異なる方法で出力画像を修正するようにしても良い。日時や場所についての関連情報を画像に自動的に掲載することにより画像をさらに特徴付けるようにしても良い。加えて、観察者の座標もまた出力画像に即座に印刷されるように、該双眼鏡をGPSに接続しておいても良い。加えて、被写体の場所を特定するための距離測定装置の提供や、該被写体に関する情報の前記出力画像への表示のような他のオプションが行われるようにしても良い。

10

技術58の説明

技術59の説明

20

【1824】

さらなる改善では、開示された技術が、カード表面にフォールト・トレラント・データ配列を与えるために利用される。該データは、他のフォールト・トレラント配列に書き換えることによりアップデートされる。

【1825】

まず、図317を参照すると、カード表面にデータ（例えば、7102）が書き込まれて、完全にデータ配列が書き込まれた後のカード1701が示されている。各データ配列（例えば、7102）は、カード表面に示されている64個のデータブロックの配列で、約20KBの情報が符号化される。もちろん、他の配列は、各データブロック（例えば、7102）の中に書き込まれるデータ量をより少なくすることも、より多くすることも可能である。初期状態では、該カード7101はデータブロックは有していない。代わりに、多分、1つの利用データブロックがそのカードの表面に書き込まれている。

30

【1826】

図318を参照すると第2のカード7105が示されており、該カードでは、書き込まれた最後のデータブロックであるところの最新のデータブロック7106で12回“アップデート”されている。したがって、各カード（例えば、7105）は、該カードに与えられたデータブロック数に従ってアップデートされ、何回も使用される。

【1827】

図319を参照すると、1つのデータブロック（例えば、7108）の構成が示されている。該ブロックは、“アートカム”技術の縮小バージョンである。そのデータブロック7108はデータエリア7109からなり、該エリアは、印刷されたドット配列を有し、該ドットは、縁に沿った一連のクロック・マークに加えて、1画素幅のボーダーを有する。そのデータエリア7109のエッジに沿ってターゲット7110が配置され、該データエリア7109の位置決め及び読み取りをアシストするようになっている。各ターゲット7110の構成は図320に示される通りであって、たった一つの白のドットを囲む、大きな黒エリアからなる。もちろん、他の構成も可能である。該ターゲット7110はデータエリア7109を正確に位置決めするために提供される。さらに、それらのターゲット7110は、さらに、正確に検知され、正確な位置情報を導き出すために提供される。

40

【1828】

図321を参照すると、カード読み取り及び書き込み装置7120が示されている。図

50

3 2 1の要点部分の拡大は図3 2 2に示されている。該装置7 1 2 0は、情報が印刷されるカード7 1 2 2を挿入するための細長い隙間7 1 2 1を有している。多くのピンチローラー（例えば、7 1 2 3, 7 1 2 4）が、プリントヘッド7 1 2 5やリニアCCDスキャナー7 1 2 6を横切るようにカードの移動を制御する。該スキャナー7 1 2 6は、下を通過するデータをスキャンする。図3 2 2には、挿入されたカード7 2 2のためのプリントヘッド7 1 2 5及びCCDスキャナー7 1 2 6の配列の拡大図が示されている。

【1 8 2 9】

図3 2 3を参照すると、CCDスキャナー7 1 2 6へカード7 1 2 2が挿入される様子が概略的に示されている。CCDスキャナー7 1 2 6は、該スキャナーを通過するようにカード7 1 2 2を搬送し、カード7 1 2 2に保存されている情報が、CCDスキャナー7 1 2 6に取り付けられているアートカム中央プロセッサユニットによってデコードされる。カードリーダーからカード7 1 2 2を取り出したい場合、新たなブロックのカードへの書き込みが必要か否かの決定がなされる。新たなブロックの位置は前にスキャンされたCCDデータから分る。カードの取り出しは、カードをプリントヘッド7 1 2 5及びCCDスキャナー7 1 2 6を通過するように、図3 2 4に示されるようにして行われる。スキャナー7 1 2 6はカード7 1 2 2の現在位置をモニターしており、図3 2 5に示すように、四角の領域7 1 2 8を新しいデータでアップデートすることが望まれる場合は、プリントヘッド7 1 2 5によりインク滴を吐出する。そして、フロッピーディスクなどと似たような方法でカードは装置(図3 2 1)から取り出される。そして、そのカードは、全てのデータスペースが埋め尽くされて新たなカードが作成されるまで、継続的に使用される。

【1 8 3 0】

それゆえ、カードを安価に作成し、情報配置のため弾力的な方法で利用可能であるという点で、前述のカードシステムの利用は、情報配置の効果的で安価な形式を提供するといことが明白だろう。

【1 8 3 1】

図3 2 6を参照すると、アートカード読み取り装置に利用される異なる機能のブロック図が示されている。プリントヘッド7 1 2 5やリニアCCD7 1 2 6は、適切にプログラムされたアートカム中央プロセッサ（ACP）チップ（ACP）7 1 3 0の制御により操作される。ACP7 1 3 0は、スキャンされたデータや他のデータ及びプログラムを保管するメモリー7 1 3 1を備えている。さらに、ACPは、ピンチローラーを作動させる種々のモータ7 1 3 2を制御する。もちろん、要求に応じて他の制御ボタン等も配置され得る。そのような編成の利用は、カード表面に格納された情報のモニタリングやアップデートのためのシステムを提供する。

技術60の説明

【1 8 3 2】

さらなる改善では、ユーザの要求に応じて自身のアートカードを作成できるアートカード写真自動販売機が提供される。該自動販売機は通常の写真自動販売機と同様の方法で構成されるが、与えられた時間内にアートカードを自動作成するように構成されたコンピュータ・リソースの要求のみが本質的要素ではない。

【1 8 3 3】

まず、図3 2 7を参照すると、さらなる改善7 2 0 1の機能要素が図解的に示されている。さらなる改善7 2 0 1は、ユーザの見本写真7 2 0 3をスキャンするための高解像度スキャナー7 2 0 2を有している。また一方、そのスキャナー7 2 0 2をオプションとして、付加された度合いのリアルリズムをアートカード作成システムに与えるため、ユーザが彼らの写真を処理することを許容しても良い。スキャナー7 2 0 2によってスキャンされた写真7 2 0 3は、適正なオペレーティングシステムとプログラムを有するハイエンドのPCタイプのコンピュータである基幹コンピュータシステム7 2 0 4に転送される。該コンピュータ7 2 0 4は、スキャンされた写真のストレージ、並びに写真プリンター及びアートカードプリンター7 2 0 5のコントロールをする。写真プリンター及びアートカードプリンター7 2 0 5は、好ましくは出力されたプリントメディアの両面に印刷すること

ができ、前に開示したような印刷技術を使用することができる。該プリンター 7205 は、アートカード 7207 に加えてユーザの写真 7206 の処理されたものを出力する。アートカードは両面が印刷される。写真 7206 を生成するため、写真 7203 の画像処理のための符号化された指示を含む。該符号化された指示はアートカードフォーマットに出力される。

【1834】

該コンピュータシステム 7204 は、また、標準的なタッチ・スクリーン・タイプのユーザ・インターフェース 7209 を有している。該コンピュータシステム 7204 は、また、支払いシステム 7210 をコントロール又は内蔵している。該支払いシステム 7210 は、標準的なコイン又は紙幣の支払いシステム、或いは、取引の承認のためのクレジットカードのサービスプロバイダと適切なネットワーク接続をするクレジット支払いシステムより成る。加えて、エフトpos (EFTPOS) 設備が提供されるかも知れない。

10

【1835】

ユーザが彼らの写真 7203 をスキャナースロット 7202 に挿入すると、写真はスキャンされ、保管され、続いて排出される。次に、“フォト・ブース”内に位置するユーザ・インターフェース 7209 をユーザが操作する。該ユーザ・インターフェース 7209 は支払いシステム 7210 に金を支払うための指示を最初に行う。しかしながら、該ユーザ・インターフェース 7209 の中核は、タッチスクリーンによる様々なアートカードの創作である。

20

【1836】

好適なユーザ・インターフェースの一例は、図 328 に示される。該ユーザ・インターフェースの中核は、後に述べる方法で処理される多くの見本サムネイル画像 7213 をユーザに示すことで構成される。最初に、例えば、会社のイベントや、誕生日のイベントや、季節のイベントや、処理のタイプ等のような画像の処理 7213 がサブジェクトエリアに配置される。ユーザは、タッチスクリーンによって好みの画像（例えば、7213）を選択する。好みの画像が存在しない場合、ユーザは矢印ボタン 7214 を押して、他の処理を表示させることができる。ユーザが画像 7213 を選択したとき、その画像に似た“テーマ”を有して、一連の処理を提示する代替物の生成にその画像が使用される。そのユーザは、選択された種類から画像を選択することを継続できる。

30

【1837】

好みの画像はセーブボタン 7217 の使用によりセーブすることができる。そして、そのユーザ・インターフェースは、セーブされている選択 7218 を示す。ナビゲーション・ボタン 7215 は前画面に戻るためのものであり、ナビゲーション・ボタン 7216 は上位に戻るためのものである。この方法によれば、ユーザは、彼ら自身の特有のカスタマイズした要求を作るため、様々なアートカードを検索することができる。この方法によれば、独特の芸術的創作活動が、独特のアートカードの作成を促す。

【1838】

そのような様々なアートカードの作成は、作成プロセスにおける創作者の役割をユーザに与える遺伝的アルゴリズム技術に依存している。ここで、図 329 を参照すると、図 327 のコンピュータシステムにて実行されるアプリケーションのソフトウェア・レイアウトの例が示されている。該ソフトウェア・レイアウト 7220 は、実行できる画像処理の遺伝的プール 7221 と、特定の画像に適合させるテーマ処理とを備えている。この遺伝的プールは、新たな種類（属）の生成のための遺伝的アルゴリズム・コアによって利用される。遺伝的アルゴリズムの標準的なテキストを紹介すると、例えば、この分野において最新の処理が加えられている“遺伝的アルゴリズム”Golberg 著がある。或いは、遺伝子のプログラミングの分野は利用され得、そして、この点で、“遺伝子プログラミング”と題名づけられたKozalによる一般的な研究を参照する。

40

【1839】

前記遺伝的アルゴリズムのコアは、また、新たに好適な画像を作成する際に、ユーザの

50

選択 7 2 2 3 を使用することができる。該遺伝的アルゴリズムのコアの出力は、タッチスクリーンディスプレイに表示するためにユーザインターフェースモジュール 7 2 2 5 に転送される、適切な新しい画像を含む。

【1 8 4 0】

再び図 3 2 8 に戻ると、ユーザが好みの選択を発見すると、受け付けボタン 7 2 1 9 を押すことができる。これにより、ユーザによる支払いが要求されるとともに受け付けられる。また、アートカード 7 2 0 7 を受け入れる他の装置（例えば、上述したアートカム装置）でユーザが利用できるように、アートカード 7 2 0 7 には、一連の画像処理に加えてプリンタ 7 2 0 5 に画像を出力させる指示がコンピュータシステム 7 2 0 4（図 3 2 7）によりなされる。

10

【1 8 4 1】

したがって、ユーザが別々に利用することのできる、複雑な、自分の名前を記入するようにカスタマイズされた画像を作成するシステムがさらなる改善によって提供されることは、当業者には明らかだろう。多くの画像の利用は、同様に重要な個性化に導くことのできるアートカードの重要な結合の急変動をもたらす。

【1 8 4 2】

もちろん、多くの他のタイプの中核技術が画像の構築に使用されることができる。例えば、他の非遺伝的技術が適するかも知れない。最悪の場合、各画像の選択は手動で用意される。さらに、他のユーザインターフェース機構が提供され得る。加えて、本発明は、標準のコンピュータシステムのような、自動販売機で無い環境で実行されることもできる。

20

【1 8 4 3】

更なる改良の操作は、個人の創造性の重要な表現に備える。そして、芸術家による個々の創造が個々の芸術家等のアートカード“シリーズ”を提供する際に重要な価値を呈するかもしれないと考えられる。

技術 6 1 の説明

技術 6 2 の説明

【1 8 4 4】

上述したアートカム配列の基本は、概略図の形態で図 3 3 0 に示されている。この配列は、画像やシーンを検知するための CCD センサー 7 4 0 2 を有する。加えて、検知画像を処理するための画像処理アルゴリズムであってコード化されたものについての、アートカード 7 4 0 8 の読み取りを行うために、アートカード読み込みセンサー 7 4 0 3 が提供される。該 CCD センサー 7 4 0 2 と該アートカード読み込みセンサー 7 4 0 3 の両方は、検知した画像の処理のために複雑な計算能力を与えるアートカード中央処理ユニット（ACP）7 4 0 4 に接続されている。加えて、メモリーユニット 7 4 0 5 が、画像や、検知したデータや、プログラムなどを保存しておくために用意されている。ACP 7 4 0 4 に接続されているのは、内部のプリントロールから供給されるプリントメディアに最終的な写真 7 4 0 4 を印刷するためのプリントヘッド 7 4 0 6 である。

30

【1 8 4 5】

さらなる改善では、独特な一連のアートカード 7 4 0 8 がアートカード・リーダー 7 4 0 3 に挿入されることにより、アートカム中央プロセッサ 7 4 0 4 の独特のコントロールがなされる。第 1 の例は、図 X 2 や図 3 3 2 に示されるものであって、重ね合わせた、或いは増大する画像の効果を与えるためのマルチプル・アートカードの使用を提供する。適切な複製アートカードは、図 3 3 2 に 7 4 1 0 で示されたものであって、効果を生じさせるためにどのようにカメラ装置を操作するかに関しての指示が一側面に設けられている。他側の面には、その効果を生じさせるためのアートカム装置の操作の指示が設けられている。図 3 3 1 を参照すると、組み合わせた効果を生じさせるためのリピティションカードの操作の例が示されている。アートカムシステムは、あるシーンの検知画像又は保存画像を有している。第 1 のステップは、リピティション・モードにするため、アートカムシステムの変更するコードを有するリピティションカード 7 4 1 3 を挿入すること

40

50

である。続いて、第1のアートカード7414がアートカード読み込みセンサー7403に挿入され、そのカード7414の指示に従って画像に第1の効果が与えられる（符号7415参照）。続いて、リピティションカードが再び挿入され（符号7416）、次に、第2のアートカード7417が挿入され、例えば、画像にテキストメッセージを配置するという第2の効果が与えられる（符号7418）。次に、再びリピティションカード7419が挿入され、その後、第3のアートカード7420が挿入されて、さらなる効果が画像7421に与えられる。図331のプロセスは、所望の出力画像の形成のため、要求に従って繰り返し行われる。この方法によれば、1つの効果だけを有するアートカードから組み合わせの効果を作成するため、柔軟性を持って上述した装置が使用される。さらに、提供されるユーザ・インターフェースは、組み合わせの効果を作成する上において、シンプルで効果的である。もちろん、多くの変更を行うことができる。例えば、別の実施例においては、リピティションカードは1回だけ挿入され、一連のアートカードがそのリピティションカードに続いて挿入される。

10

【1846】

図333を参照すると、アートカムシステムの内部テストのための、別のアートカード7430が示されている。一つ一つのアートカムシステムには多くの内部テストルーチンが内部ROMに用意されている。そのテストは、アートカム中央プロセッサに用意された翻訳言語での、特別な関数呼出しによりアクセスされる。そのルーチンは、アートカム装置の仕様であり、例えば、

20

プリントヘッドの動作状態を判断するためのテストパターンのプリントアウトと；
ノズルのクリーニングのため、及びプリントヘッドの操作が改良されるようなノズル配列のため、プリントヘッドの操作処理を行うテストパターンのプリントアウト（例えば、全てが黒のプリント）と；

プリントヘッドの操作の有効性を示すため、後の分析のためにテストパターンがプリントされること；

が含まれる。

【1847】

図334を参照すると、テストパターン7437のプリントアウトに加え、参考になる様々な内部データを有するテスト・アウトプット7435の見本が示されている。そのテストパターン7437は、そのカメラシステムに存在しているかも知れない何らかの問題を割り出す自動的又は主導的な方法によって後で調査することができる。さらなる改善は、アートカム装置にプログラムされ、ROMメモリーに格納されているソフトウェア・ルーチンの使用を通じて実行される。

30

【1848】

もちろん、前記ルーチンがアップデートされたり、変更されたりすることで、多くの改善が予想され得る。多くのテストは実際には制限されない。この方法によれば、カメラ装置の操作は、挿入されるカードに従って変更され得る。

技術63の説明

技術64の説明

【1849】

40

さらなる改善では、出力写真の裏面に音声メッセージを記録するために磁気感受性プリントメディア材料が使用される。アートカム装置は、写真の全面を覆う磁気レコーダの配列や、或いは、例えば、磁气的感受性のある写真中央部のような、磁気帯からなる磁気記録装置を含むように改変される。該アートカム装置にはさらに後で再生するための音声メッセージを録音できる能力が与えられる。

【1850】

さらなる改善では、アートカム装置は、マイクロフォン装置や関連の録音技術を装備するように適当に変更される。写真を撮影するとき、周囲の音環境か、その画像に関係するメッセージのいずれかを録音する機会が与えられる。そのプリントメディア又はフィルムは、テープメディアと同じように、磁気感受性を有するように前処理されている。出力写

50

真の裏面全体、或いは磁気感受性帯に録音がなされる。録音された音声は、コード化された形式で出力写真の裏面に保持される。そのコード化は、好ましくは、高度なデジタル再現性のある形式である。その録音された音声は写真に関係する、永久的な音声録音を提供する。次に、再生装置は、コード化された音声をスキャンし、この情報を復号化するために提供される。

【1851】

図335を参照すると、上述した構成、つまり、画像7602がCCDセンサー7603により検知されアートカム中央プロセッサ7604に転送されるという構成を含むさらなる改善7601が、概略形式にて示されている。該アートカム中央プロセッサ7604は、好ましくは高速RAMBUS（登録商標）インターフェースのメモリーからなるメモリー7605に画像を格納する。また、アートカム中央プロセッサ7604は、写真の裏に記録をするための磁気記録ヘッド7616に加え、プリントヘッド7606（要求に応じてインスタント画像を提供するため、フルカラー写真7607を印刷するためのもの）の操作をコントロールするようになっている。

【1852】

さらなる改善では、カメラ編成7601は、ACPプロセッサ7604の制御下でRAMBUSバス7611経由でメモリー7605にインターフェース接続されているサウンドチップ7610を備えている。該サウンドチップ7610は、標準的形式とすることも、或いは専用形式とすることもでき、例えば、サウンドマイクロフォン7613からのアナログ入力7612を取得するDSPプロセッサを備えることができる。もう一つの方法として、チップの複雑さに伴って（ムーアの法則）、サウンドチップ7610の機能を、好ましくは最先端のCMOS型集積回路チップを備えるACPチップ7604に組み込むことができる。本発明の範囲内に該当する他の多くの種類の編成を提供できることは容易に明白であろう。

【1853】

サウンドチップ7610は、アナログ入力7612を対応するデジタル形式へ変換し、それをメモリー7605に格納するために転送する。そのような記録プロセスは、カメラ装置上のボタン（図示せず）を押すことによって作動させることができる。該ボタンはACPプロセッサ7604の制御下にあるか、そうでなければ、写真を撮るときに実質的に自動化されることができる。記録されたデータは、メモリー7605に格納される。

【1854】

図336を参照すると、カメラ編成は、好ましくは、プリンター装置7606（プリントメディア7617に画像を印刷するのに使用されるプリントヘッドを有する）と磁気記録プリントヘッド7616（プリントメディア7617の裏面に情報を印刷するのに使用される）とを備えている。図337を参照すると、フォトメディア7617の裏面に形成された磁気帯7618の出力例が示されている。その磁気帯は図336の記録ヘッド7616によって記録される。記録された情報は、場所や日付や時間のデータを含み、場所のデータはキーボード入力により、或いは、GPSや同様のもののような、付属の位置決めシステムの利用によって与えられる。重要なことには、画像7617の裏面に、コード化された音声情報が記録される。コード化の方法には多くの様々なものとすることができるが、好ましくは、エラーを許容するように、高度なフォールトトレラント方法でコード化される。その高度化の形式は、高度なフォールトトレラントを提供するため、データのリード・ソロモン符号化を利用するようにしても良い。

【1855】

図338を参照すると、録音された音声を“再生”したいときは読み取り装置7626を用い、ピンチローラーにて写真7617を挟み込みながら磁気センサー装置7627を通過させる。

【1856】

図339を参照すると、図338の音声読み取り装置7626の操作が概略形式で示されている。前記磁気センサー7627は第2のアートカム中央プロセッサ7628に接続

されていて、写真の裏面に格納されているデータを読み取って復号化するようになっている。その復号化された音声情報はメモリー７６３２に格納され、サウンド処理チップ７６３３を介してスピーカー７６２９で再生される。そのサウンド処理チップ７６３３はＡＣＰデコーダー７６２８の制御下で操作され、ボリュームコントロールや巻き戻しや再生や早送り等の、ユーザの様々な入力制御７６３３に基づき順に操作される。

【１８５７】

さらなる改善についての上記説明から理解できるように、出力画像に関係する音声の自動録音のためのシステムが提供され、写真に関係する音声録音が行われる。また、写真の裏面に記録された画像を読み取るための音声読み取りシステムも開示される。数多くの変更及び／又は修正が、大まかに説明された本発明の精神または範囲から逸脱することなく、特定の実施の形態に示されるような本発明へなされてもよいことは、当業者によって理解されよう。例えば、ステレオおよびＢフォーマット技術等のような、より複雑な音声録音再生技術への利用である。

10

技術６５の説明

【１８５８】

さらなる改善では、プリントヘッドを使用して“ネガ”を印刷するよう、上述したアートカム装置が適正にプログラムされる。該ネガは、アートカード・センサー装置に挿入されて利用できるよう、標準的なアートカード形式に適合している。アートカード・センサー装置は“ネガ”アートカードに格納された情報を検知し、写真を印刷する。

【１８５９】

20

まず、図３４０を参照すると、要求に応じて画像を検知するセンサー装置７７０２と、アートカード情報を検知するためのカード読み取りセンサー７７０３とを備えた標準的なアートカム装置７７０１が、概略形式で示されている。それらのセンサー７７０２、７７０３は、アートカム装置の操作のための重要な計算手段であるアートカム中央プロセッサ７７０４の制御下で操作される。そのアートカム中央プロセッサ７７０４は、完全にベクトル化されたＶＬＩＷ（very long instruction word）中央プロセッサに加えて、オンボードＣＰＵを含むことができる。ＡＣＰ７７０４は、取得した画像７７０２の処理に加えて、アートカード読み取り装置７７０３にて読み取ったアートカード（情報）の処理及び復号化を行う。内部プリントロールにより供給されるプリントメディアにプリントヘッド７７０６により印刷する処理画像を形成し、写真７７０７を出力するため、ＡＣＰ７７０４はメモリー７７０５との間での処理を行う（interact）。該アートカムシステム７７０１は、完全なポータブルカメラシステムを構築するように設計されており、プリントヘッド７７０６（該プリントヘッドは、ＡＣＰプロセッサ７７０４の制御下において要求により写真７７０７の印刷を行うためのものである）に接続されたプリントロール（内蔵型で使い捨てできるものであって、印刷メディア及びインクを供給するためのものである）を備えている。

30

【１８６０】

図３４１を参照すると、図３４０のアートカムシステムの操作のステップであって、さらなる改善に従ったもの７７１０が示されている。さらなる改善では、まず、第１のアートカム装置を使って所望の画像が撮影される（７７１０）。次に、高度なフォールトトレラントであって、回復される出力データ形式を提供すべく、該撮影画像が処理される。その処理は、画像、複製されたデータ、データ値の配列にデータを疑似ランダムに展開するのに加えてリード・ソロモン符号化データの圧縮を含むことができる。その出力形式は、必要なら、画像の圧縮に加え、上述したような“アートカード”装置により提供されるのと同じにできる。アートカードの出力データに加え、検知画像のサムネイル画像がその符号化されたデータに沿って印刷される。

40

【１８６１】

そのアートカード編成は、図３４２に示されるように、ＣＣＤセンサー７７０２によって画像７７２０を取得すると共に、アートカムの“ネガ”のもの７７２１を出力するアートカム装置７７０１を備えている。該“ネガ”形式のものは、入力画像のサムネイル７７

50

24と、検知画像を高度に符号化したもの7722とを含む。図342には示されていないけれども、その出力されたものは、アートカム装置7701内の裁断装置によって線7725に沿って通常は切り取られる。

【1862】

データエリア7722は、該エリア7722の側縁に沿った一連の印（例えば、7727, 7728）と共に印刷される。それらの印7727, 7728は、スキャンされるときにデータエリア7722の正確な位置を与える。“ネガ”7721を印刷することにより、画像の半永久的な記録が得られる。この“ネガ”7721は、写真のネガを保存するのと同様の方法により保存される。取得した画像の、さらなる、高品質なコピーを必要とする場合には、前記アートカード“ネガ”が、図342のアートカム装置のアートカード読み取り装置7703の中へ挿入され（図341の7713）、アートカードデータは通常の方法で復号化され、オリジナル画像が得られる。符号化されているデータは、オリジナルな画像と、該画像をアートカム装置により印刷するために必要な指示を含むことができる。検知されたカードは復号化され（図341の7714）、要求に応じて何度も印刷され（7715）、オリジナル画像が得られる。つまり、検知された画像が出力される。よって、検知された画像を高解像度と高い回復力でコピーするシステムが得られる。

【1863】

他の実施例も構成することができる。例えば、前記“ネガ”データを、写真面の裏面に自動印刷する編成を利用しても良い。そのような編成は図343に示される。該図に示されるものは、写真7730の裏面であり、サムネイル部分7732に加えて、データ符号化エリア7731が印刷されている。したがって、該データエリア7731は、該カード7730の表面において、画像に関しての、半永久的で、高い精度の記録を提供することができる。

技術66の説明

【1864】

さらなる改善では、ランダムな出力の要求に応じて多数のアートカードを出力できるアートカード・プリンターが提供される。その製品のプリンターは、ネットワークを介して他の製品のプリンターと接続され、総合コンピュータシステムが、所定の要求に従ってアートカードを分配するためにネットワークを制御する。このような方法で、高い柔軟性を与えるため、アートカードの製造がコントロールされる。

【1865】

図344を参照すると、たった一つのアートカードプリンター7801が概略図の形式で示されている。該プリンターは、アートカードの上面に画像を印刷するためのプリントヘッド7802、及びアートカードの下面に画像取扱説明書（符号化されたもの）を印刷するためのプリントヘッド7803の2つのプリントヘッド7802, 7803を備えている。該プリントヘッド7802はインク供給部7806を有する3色のプリントヘッドにすることができる。下面プリンター7803は1つのインク供給部7807を有する単色プリントヘッドにすることができる。プリンター7801は、要求に応じてアートカードをプリントアウトすべくコンピュータシステムの制御にて操作される。印刷した後、ベースプレート7810（流通目的のために多数のアートカードが配置されたもの）からアートカードを打ち出すためにパンチ装置7809が配置されている。所定の区域の要求に従った分配。

【1866】

図345を参照すると、アートカード交付システムの制御ネットワーク7820の一般的な配列が示されている。一連のプリンターコンピュータ装置（例えば、7821）が配置され、要求に応じて様々な形態を取ることのできるネットワーク7822と一緒に接続されている。中央統合コンピュータ7823がネットワーク7822及びプリンターコンピュータ（例えば、7821）を制御して、プリンターコンピュータ（例えば、7821）によってアートカードの組み合わせたものを印刷する指示を送るようになっている。プリンターコンピュータ（例えば、7821）は、要求に応じて地方に配置することができ

る。或いは、それらのプリンターコンピュータを中心（centrally）に配置しても良いが、様々な地方に送られ、或いは、様々な要求を満足するために命令されるかも知れない。それぞれのプリンターコンピュータは、アートカードの大きなキャッシュを有することができる。そのキャッシュは、メモリーや、ディスクや、各プリンターコンピュータに適宜配布されるC D－R O Mに設けることができる。中央統合コンピュータは、プリンターコンピュータによって印刷されるべきアートカードのリストを配布する。加えて、新しいアートカードが該中央統合コンピュータ7832に読み込まれ、要求に応じ、関連するプリンターコンピュータ（例えば、7821）に送られる。その配布は、キャッシュされた配布配列の形式である。

【1867】

図346を参照すると、プリンターコンピュータの好適な編成7830の1つの形態が示されている。その編成7830は基幹コンピュータシステム7831を備え、該コンピュータシステム7831は、大きなメモリーやディスクキャパシティを持つ、ハイエンドのP Cタイプのコンピュータである。そのコンピュータシステムは7831は、図344の編成に従って配列されたアートカードプリンター7832に接続されている。該コンピュータシステム7831は、ネットワーク7834に加えて、キャッシュされるアートカード7833の大きなストアを備えている。よって、プリントアウトされるべきアートカードのリストは前記中央コンピュータから各プリンターコンピュータへ、アートカードの印刷のために転送される。

【1868】

図345の編成では、大きな組み合わせの複雑な配布の問題が打開され、配布のシステムの改良された形態であることが分る。

インクジェット技術

【1869】

本発明の実施例はインクジェットプリンタ型装置を使用している。もちろん、多くの異なる装置を使用することができる。しかしながら、現在ポピュラーなインクジェットプリント技術は適していそうにも無い。

【1870】

サーマルインクジェットでの最も重要な問題は電力消費である。インク滴の射出のエネルギー効率が悪いことに起因し、高速のために必要な電力は約100倍である。インクを吐出させる蒸気泡を生成するために、水の迅速な沸騰を必要とするからである。水は大変高い熱容量を有していて、サーマルインクジェットの利用において過熱される。これは、電気の入力を運動の出力に変換するため、約0.02%の効率を必要とする。

【1871】

ピエゾエレクトリックインクジェットでの最も重要な問題はサイズとコストである。ピエゾエレクトリッククリスタルは適当な駆動電圧にて非常に小さなデフレクション（deflection）を有していて、それ故、各ノズルのために大きなエリアが必要となる。また、各ピエゾエレクトリックのアクチュエータは、分離した基板の駆動回路に接続されなければならない。このことは、300程度のノズルの電流制限においては重要な問題ではないが、19,200のノズルを有するページ幅プリントヘッドの製造には大きな障害となる。

【1872】

理想としては、使用されたインクジェット技術は、非公開のデジタルカラープリンティングや、他のハイクォリティでハイスピードでローコストのプリント利用が要求される。デジタル写真の要求に合わせるため、新たなインクジェット技術が創出されてきた。目的の特徴は次のものである。：すなわち、

低電力（10ワット未満）

高解像度性能（1,600dpi又はそれ以上）

写真クォリティの出力

低製造コスト

小サイズ（横切り幅が最小となるようにページ幅を調整する）
ハイスピード（＜1 頁当たり 2 秒）

【1873】

これらの特徴の全ては、後述するインクジェットシステムにより、異なるレベルの困難さで突破されることが可能である。45 個の異なるインクジェット技術は、高ボリュームの製造のために幅広い選択を与えるよう、受け継ぐ者によって発展されてきた。これらの技術は、後に記載する表に示すように、本出願人に指定された分離された応用を形成する。

【1874】

ここに示されるインクジェットの設計（デザイン）は、電池で駆動される 1 回使用のデジタルカメラから、卓上のネットワークプリンタや業務用のプリントシステムまで、幅の広いデジタルプリントシステムに適している。

【1875】

標準の装置を使用して簡単に製造するために、前記プリントヘッドは、MEMS 後処理法によって、モノリシックの 0.5 ミクロンの CMOS チップに設計される。カラー写真の応用のため、前記プリントヘッドは 100 mm の長さで、インクジェットのタイプに応じた幅を有する。最小のプリントヘッドは IJ38 であり、幅は 0.35 mm で、 35 mm^2 のチップ面積を有する。そのプリントヘッドは 19,200 個のノズルとデータ及び制御回路を有している。

【1876】

インクは、射出成形されたプラスチック製のインク通路を経由して、プリントヘッドの背面に供給される。そのモールドイングは 50 ミクロの特徴（features）が必要とされる。その特徴は、標準的な射出成型工具の中にリソグラフィでマイクロマシン加工されたインサートを使用して形成されることができる。インクは、ウエハの正面に形成されたノズル・チャンバーへ、ウエハを貫通するように形成された孔を通して流れる。そのプリントヘッドは TAB によりカメラ回路に接続されている。

相互参照付きのアプリケーション

【1877】

次表は、最近出願された米国特許出願のガイドである。それらの出願は、これと共に一斉に提出され、特別なケースに言及するときに、その後の表で使用される参考を用いて検討されている。これらの出願は、また、対応する参考番号を有するオーストラリア仮特許出願（前述の表にて示されたような）のように出願されている。

10

20

30

【表 2 1 2】

整 理 番 号	参考	タイトル
IJ01US	IJ01	ラジアルプランジャのインクジェットプリンタ
IJ02US	IJ02	静電式インクジェットプリンタ
IJ03US	IJ03	プレーナー熱弾性曲げアクチュエータインクジェット
IJ04US	IJ04	積み重なった静電式インクジェットプリンタ
IJ05US	IJ05	逆のスプリングレバーインクジェットプリンタ
IJ06US	IJ06	パドルタイプのインクジェットプリンタ
IJ07US	IJ07	永久磁石・電磁石インクジェットプリンタ
IJ08US	IJ08	プレーナスウィンググリル電磁石インクジェットプリンタ
IJ09US	IJ09	ポンプアクション補給インクジェットプリンタ
IJ10US	IJ10	パルス化された磁場のインクジェットプリンタ
IJ11US	IJ11	2枚板の逆ファイアリング電磁石インクジェットプリンタ
IJ12US	IJ12	リニアステッパアクチュエータインクジェットプリンタ
IJ13US	IJ13	ギヤ駆動のシャッターインクジェットプリンタ
IJ14US	IJ14	先細のマグネチックポール電磁石インクジェットプリンタ
IJ15US	IJ15	線形スプリング電磁石グリルインクジェットプリンタ
IJ16US	IJ16	ローレンツダイヤフラム電磁石インクジェットプリンタ
IJ17US	IJ17	P T F E 面シューティングシャッタード圧力発生インクジェット プリンタ
IJ18US	IJ18	バックルグリル圧力発生インクジェットプリンタ
IJ19US	IJ19	シャッターベーストインクジェットプリンタ

10

20

30

【表 2 1 3】

IJ20US	IJ20	カーリングカリックス熱弾性インクジェットプリンタ
IJ21US	IJ21	熱アクチュエータインクジェットプリンタ
IJ22US	IJ22	アイリスモーションインクジェットプリンタ
IJ23US	IJ23	直火型熱曲げアクチュエータインクジェットプリンタ
IJ24US	IJ24	導電性 P T F E 曲げアクチュエータインクジェットプリンタ
IJ25US	IJ25	磁気歪みインクジェットプリンタ
IJ26US	IJ26	形状記憶合金インクジェットプリンタ
IJ27US	IJ27	バックルプレートインクジェットプリンタ
IJ28US	IJ28	熱弾性ロータリインペラインクジェットプリンタ
IJ29US	IJ29	熱弾性曲げアクチュエータインクジェットプリンタ
IJ30US	IJ30	P T F E 及び銅波板使用の熱弾性曲げアクチュエータのインクジェットプリンタ
IJ31US	IJ31	インクを直接供給する曲げアクチュエータのインクジェットプリンタ
IJ32US	IJ32	高ヤング率熱弾性インクジェットプリンタ
IJ33US	IJ33	チャンバー壁が熱的に駆動されるインクジェットプリンタ
IJ34US	IJ34	外部のコイルスプリングを諷する熱アクチュエータを備えたインクジェットプリンタ
IJ35US	IJ35	桶状容器を有するインクジェットプリンタ
IJ36US	IJ36	デュアルチャンバー・シングル垂直アクチュエータインクジェット
IJ37US	IJ37	デュアルノズル・シングル水平支柱アクチュエータインクジェット
IJ38US	IJ38	デュアルノズル・シングル水平アクチュエータインクジェット
IJ39US	IJ39	パドルが凹まされたシングル曲げアクチュエータのインクジェットプリンタ装置
IJ40US	IJ40	一連の熱的アクチュエータユニットを有する、熱的に駆動されるインクジェットプリンタ
IJ41US	IJ41	先細型のヒータ部材を有する熱的に駆動されるインクジェットプリンタ
IJ42US	IJ42	放射状バックカールの熱弾性インクジェット
IJ43US	IJ43	反転される放射バックカールの熱弾性インクジェット
IJ44US	IJ44	インクを放出する表面曲げアクチュエータのインクジェットプリンタ
IJ45US	IJ45	磁石板を駆動するコイルのインクジェットプリンタ

10

20

30

40

ドロップ・オン・デマンド方式のインクジェット

【 1 8 7 8 】

個々のインクジェットノズルの基本動作に関する 11 個の重要な特徴が特定されている。これらの特徴は大体は直角 (orthogonal) であり、したがって、11 次元のマトリクスとして解明されることができる。このマトリクスの 11 軸のほとんどは、本出願人により発展された記入事項を含む。

50

【1879】

次の表は、インクジェットタイプの11次元の表の軸を形成する。

アクチュエータ・メカニズム（18タイプ）

基本動作モード（7タイプ）

補助のアクチュエータ（8タイプ）

アクチュエータの増幅及び改良方法（17タイプ）

アクチュエータの動き（19タイプ）

ノズル補給方法（4タイプ）

吸入口への逆流を制限する方法（10タイプ）

ノズルの掃除方法（9タイプ）

ノズルプレート構造（9タイプ）

滴の噴出方向（5タイプ）

インクタイプ（7タイプ）

10

【1880】

これらの軸により表示された完全な11次元の表は、インクジェットノズルに関し、369億の可能な形態を含む。様々なインクジェット技術においてそれらの全てが実現可能ではないけれども、数百万は実行可能である。可能な形態の全てを説明することは、明らかに非現実的である。その代わり、いくつかのインクジェットタイプが詳細に吟味されてきた。それらが、上述の、指名されたIJ01からIJ45である。

【1881】

20

他のインクジェットの形態は、11軸の1又はそれ以上に沿って代替となる形態に置き換えることにより、これらの45の例から直ちに導き出すことができる。IJ01からIJ45のほとんどは、何らかの現在利用できるインクジェット技術より上位である特徴を、インクジェットプリントヘッドに作り込むことができる。

【1882】

発明者に知られている先行技術例がある場合には、これらの1又はそれ以上が、下記の表の例の欄に示される。該IJ01からIJ45のシリーズはまた、例の欄にも示されている。幾つかのケースでは、プリンタが、1つ以上の特徴を共有する場合には、1つの表に1回以上示されるかも知れない。

【1883】

30

好適な応用は以下のものを含む。すなわち、家庭用プリンタ、オフィス用ネットワークプリンタ、短期用デジタルプリンタ、業務用プリントシステム、布用プリンタ、ポケットプリンタ、インターネットwwwプリンタ、ビデオプリンタ、医療用画像、大判プリンタ、ノート型パソコン用プリンタ、ファックス機、工業用プリントシステム、写真コピー機、写真現像店等。

【1884】

前述した11次元のマトリクスで関連付けられた情報が以下の表に示される。

【1885】

【表 2 1 4】

アクチュエータのメカニズム (選択されたインク滴にのみ適用される)

アクチュエータのメカニズム	説明	メリット	デメリット	例
サーマルバブル	電熱のヒーターは、水性インクを著しく暖めて、インクを沸点より高温に加熱する。バブル(気泡)は核となって素早く形成され、インクを吐出する。プロセスの効率は低い。滴の運動エネルギーに変換される電気エネルギーは、通常は、0.05%よりも少ない。	大きな力が発生する。 製造が簡単。 可動部が無い。 動作が速い。 アクチュエータのために要求されるチップ面積は小さくて良い。	電力が大きい。 インクキャリアは水に限定される。 低効率 高温になる。 機械的歪みが大きい。 特異な材料が必要。 駆動トランジスタが大きい。 キャビテーションがアクチュエータの故障を引き起こす。 Kogation が気泡の形成を減少させる。 大きなプリントヘッドを製造するのが困難である。	キヤノンバブルジェット 1979 遠藤ら、英国特許 2,007,162 ゼロックス、ヒーター・イン・ビット、1990年、ホーキスら、USP4,899,181 ヒューレットパッカード、TIJ、1982年、Vaught ら、USP4,490,728
ピエゾエレクトリック	ジルコン酸鉛ランタン(PZT)のようなピエゾエレクトリッククリスタルが電氣的に駆動され、滴吐出のために、インクに圧力を供給するために、膨張か、share か、曲げが行われる。	電力消費が少ない。 多くのタイプのインクを使用できる。 動作が速い。 高効率	アクチュエータのために非常に大きなエリアが必要。 電気で統合(integrate)することが困難。 高電圧駆動トランジスタが必要。 アクチュエータのサイズのため、フルのページ幅プリントヘッドは実現できない。 製造の際、高電界においてエレクトロカル・ポーリングが必要。	カイザーら、USP3,946,398 Zoltan USP3,683,212 1973年、Stemm、USP3,747,120 エプソンスタイラス テクトロニクス IJ04

10

【1886】

【表 2 1 5】

20

電歪(電気ひずみ)	電場が、ジルコン酸チタン酸鉛ランタン(PLZT)やマグネシウムニオブ酸鉛(PMN)のようなリラクサー材料中で、電歪を作動させるのに使用される。	電力消費が少ない。 多くのタイプのインクを使用できる。 熱膨張が小さい 必要な電場強度(約3.5V/ μ m)は困難なく発生させることができる。 エレクトロカル・ポーリングは不要。	最大歪みは小さい(約0.01%) 歪みが小さいにもかかわらず、アクチュエータのためには大きな面積が必要 応答スピードが不十分(～10 μ s) 高電圧駆動トランジスタが必要。 アクチュエータのサイズのため、フルのページ幅プリントヘッドは実現できない。	セイコーエプソン、碓井ら、特願平8-253401 IJ04
強誘電体	電場が、反強誘電性(AFE)と強誘電性(FE)相の間の相転移を誘発するために使用される。 錫で修飾されたジルコン酸チタン酸鉛ランタン(PLZSnT)のような灰チタン石材料が、AFEからFEへの相転移で結合される1%の大きな歪みを示す。	電力消費が少ない。 多くのタイプのインクを使用できる。 動作が速い(<1 μ s)。 縦歪が比較的高い。 高効率。 3V/ μ m前後の電場強度を直ちに付与できる。	電気で統合(integrate)することが困難。 PLZSnTのような特異な材料が必要である。 アクチュエータは大きな面積が必要。	IJ04
静電プレート	導電性のプレートは、圧縮性の、或いは流体の誘電体(通常は空気)により隔てられる。電圧が印加されると、それらのプレートは引き付けられてインクを排除し、滴の吐出を引き起こす。それらの導電性プレートは、くし歯状でもハニカム状でも良い。表面積を増加することによって力を増加させるため、積み重ねるようにしても良い。	電力消費が少ない。 多くのタイプのインクを使用できる。 動作が速い。	水性の環境において静電気のデバイスを操作することが困難。 静電気のアクチュエータは通常はインクから分離されている必要がある。 大きな力を発生させるには非常に広い面積が必要。 高電圧駆動のトランジスタが必要。 フルページ幅のプリントヘッドはアクチュエータサイズが原因で性能は良くは無い。	IJ02, IJ04

30

40

【1887】

【表 2 1 6】

インクの静電吸引	強電場がインクに印加されるとすぐに、静電気力は、そのインクをプリントメディアの方に加速させる。	消費電流が少ない。 低温	高い電圧が必要。 空気の絶縁破壊に伴うスパークで損傷を受けるかも知れない。 滴のサイズが小さくなるにつれ、電場強度の増加が要求される。 高電圧駆動のトランジスタが必要。 静電場が埃を引き付ける。	1989 年、斉藤ら、USP4, 799, 068 1989 年、三浦ら、USP4, 810, 954 Tone-jet
永久磁石 電磁石	インクを移動させて滴の射出が引き起こされるよう、電磁石は永久磁石を直接引き付ける。約 1 テスラの磁界強度を有する希土類永久磁石を使用できる。例えば、サマリウム・コバルト (SmCo) や、ネオジム・鉄・ボロン系 (NdFeB, NdDyFeB, NdDyFeB 等) 中の磁性材料である。	電力消費が少ない。 多くのタイプのインクを使用できる。 動作が速い。 高効率 シングルノズルからページ幅プリントヘッドへの簡単な拡張性	製造が複雑 ネオジム・鉄・ボロン (NdFeB) のような永久磁性材料が必要。 大きい局所電流が必要。 長いエレクトロマイグレーション寿命と低い抵抗値のために銅材料が使用されるべき。 顔料インクは使用不可能である。 稼働温度はキュリー温度 (約 540 K) に限定される。	IJ07, IJ10
軟磁性コア 電磁石	CoNiFe [1] や CoFe や NiFe 合金のような電気めっき鉄合金のような磁性材料にて製造された軟磁性コアやヨーク中の磁場をソレノイドが減少させる。典型的には、軟磁性材料は 2 箇所、通常はバネにより離間された状態である箇所にある。ソレノイドが作動されたとき、その 2 つの部分は引き付けられ、インクが排除される。	電力消費が少ない。 多くのタイプのインクを使用できる。 動作が速い。 高効率 シングルノズルからページ幅プリントヘッドへの簡単な拡張性	製造が複雑 NiFe や CoNiFe や CoFe のような、通常は CMOS 製造で使用しない材料が必要とされる。 大きい局所電流が必要。 長いエレクトロマイグレーション寿命と低い抵抗値のために銅材料が使用されるべき。 電気メッキが必要。 高飽和磁束密度が必要 (2.0 - 2.1 T が CoNiFe [1] で実現される。)	IJ01, IJ05, IJ08, IJ10 IJ12, IJ14, IJ15, IJ17

10

20

【1888】

【表 2 1 7】

磁気ローレンツ力	磁界中で導電ワイヤに作用するローレンツ力が利用される。この場合の磁界は、例えば、希土類永久磁石によって、外部からプリントヘッドに与えることができる。プリントヘッドには、電流通電ワイヤのみが必要で、材料への要求は単純化される。	電力消費が少ない。 多くのタイプのインクを使用できる。 動作が速い。 高効率 シングルノズルからページ幅プリントヘッドへの簡単な拡張性	力がねじり運動を生じさせる。通常は、ソレノイド長の 4 分の 1 のみが、有益な方向に力を発生させる。 大きい局所電流が必要。 長いエレクトロマイグレーション寿命と低い抵抗値のために銅材料が使用されるべき。 顔料インクは使用不可能である。	IJ06, IJ11, IJ13, IJ16
磁歪	Terfenol-D (Naval Ordnance Laboratory で開発されたテルビウムとジスプロシウムと鉄の合金だから、TerFe-NOL) のような材料の大きな磁歪効果をアクチュエータが使用する。最高の効率のためには、そのアクチュエータは約 8 MPa にプレストレスされるべきである。	多くのタイプのインクを使用できる。 動作が速い。 シングルノズルからページ幅プリントヘッドへの拡張は容易。 強い力を利用できる。	力がねじり運動を生じさせる。Terfenol-D のような特異な材料が必要。 大きい局所電流が必要。 長いエレクトロマイグレーション寿命と低い抵抗値のために銅材料が使用されるべき。 プレストレスが必要かも知れない。	フィッシャートン、USP 4, 032, 929 IJ25
表面張力の減少	正圧下のインクは、表面張力によってノズル中に保持される。その表面張力は、限界値以下に低減され、インクはノズルから吐出される。	電力消費が少ない。 構造が簡単。 製造時に特異な材料は必要とされない。 高効率。 シングルノズルからページ幅プリントヘッドへの拡張は容易。	滴を分離するための補助力が必要。 特別なインク界面活性剤が必要。 スピードは界面活性剤の品質により限定されるかも知れない。	シルバートン、EP 0771 658 A2、及び関連特許出願

30

40

【1889】

【表 2 1 8】

粘性の減少	噴出される滴を選択するために、インクの粘性が局所的に低減される。粘性低下はほとんどのインクでは電熱的に成し遂げることができるが、特殊なインクは100:1の粘性低減を設計できる。	構造が簡単。 製造時に特殊な材料は必要とされない。 シングルノズルからページ幅プリントヘッドへの拡張は容易。	滴を分離するための補助力が必要。 特別なインクの粘性特性が必要。 高速にすることは困難。 インク圧を変動させることが必要とされる。 大きな温度差（通常は80度）が必要である。	シルバーブルック、EP 0771 658 A2、及び関連特許出願
音響	音波が発生され、滴が射出される領域に収束される。	ノズルプレートなしに駆動できる。	回路を駆動するのが複雑。 製造が複雑 低効率 滴の位置のコントロールが不得手。 滴の大きさのコントロールが不得手。	1993年、Hadimioglu ら、 EUP550, 192 1993年、Elrod ら、 EUP572, 220
熱弾性曲げアクチュエータ	ジュール加熱により熱膨張が異なるようなアクチュエータが使用される。	電力消費が少ない。 多くのタイプのインクを使用できる。 簡単なプレーナ製造である。 各アクチュエータのためのチップ面積は小さい。 動作が速い。 高効率。 電圧と電流を両立するCMOS標準的なMEMSプロセスを使用できる。 シングルノズルからページ幅プリントヘッドへの拡張は容易。	効率的な水の運用には、熱い側に断熱材が必要である。 腐食防止が難しい。 色素粒子が曲げアクチュエータを jam するかも知れないため、顔料インクは使用不可能かも知れない。	IJ03, IJ09, IJ17, IJ18 IJ19, IJ20, IJ21, IJ22 IJ23, IJ24, IJ27, IJ28 IJ29, IJ30, IJ31, IJ32 IJ33, IJ34, IJ35, IJ36 IJ37, IJ38, IJ39, IJ40 IJ41

10

【1890】

20

【表 2 1 9】

高 CTE の熱弾性のアクチュエータ	ポリテトラフルオロエチレン (PTFE) のような、非常に高い熱膨張率 (CTE) の材料が使用される。高 CTE 材料は通常は非導電性なので、導電材料から作られたヒーターが組み込まれる。ポリシリコンヒーターや15mWの電力投入で、50μm長の PTFE の曲げアクチュエータが、180μNの力や10μmのデフレクションを与えることができる。アクチュエータの動きは次のようなものである。 1) 曲げ 2) 押し 3) バックル 4) 回転	大きな力を発生させることができる。 PTFE は、ULSI において低い誘電率の絶縁材のための候補である。 電力消費が非常に少ない。 多くのタイプのインクを使用できる。 簡単なプレーナ製造。 各アクチュエータのためのチップ面積は小さい。 動作が速い。 高効率。 電圧と電流を両立するCMOS。 標準的なMEMSプロセスを使用できる。 シングルノズルからページ幅プリントヘッドへの拡張は容易。	特別な材料が必要（例えば、PTFE） ULSI の製造では未だ一般的でない、PTFE の蒸着法が必要である。 PTFE の蒸着は、350℃以上の高温では不可能である。 色素粒子が bend actuator を jam するかも知れないため、顔料インクは使用不可能かも知れない。	IJ09, IJ17, IJ18, IJ20 IJ21, IJ22, IJ23, IJ24 IJ27, IJ28, IJ29, IJ30 IJ31, IJ42, IJ43, IJ44
導電性高分子熱弾性アクチュエータ	PTFE のように、高い熱膨張率を有するポリマーが、銅の導電性よりも約3桁小さくなるように伝導性が増加されるように、導体物質にドーパされる。その導電性ポリマーは抵抗発熱に伴い膨張する。 導体ドーパンの例は次のものを含む。 1) カーボン・ナノチューブ 2) 金属繊維 3) ドープされたポリチオフェンのような導電性ポリマー 4) 炭素粒	大きな力を発生できる。 電力消費が非常に少ない。 多くのタイプのインクを使用できる。 簡単なプレーナ製造。 各アクチュエータのためのチップ面積は小さい。 動作が速い。 高効率。 電圧と電流を両立するCMOS。 シングルノズルからページ幅プリントヘッドへの拡張は容易。	特別な材料の発展が必要である（高CTE導電性ポリマー） ULSI の製造では未だ一般的でない、PTFE の蒸着法が必要である。 PTFE の蒸着は、350℃以上の高温では不可能である。 蒸発やCVDの蒸着法は使用できない。 色素粒子が bend actuator を jam するかも知れないため、顔料インクは使用不可能かも知れない。	IJ24

30

40

【1891】

【表 2 2 0】

形状記憶合金	TiNi (また、Naval Ordnance Laboratory で開発されたニッケル・チタン合金—ニチノールとして知られている) が、弱いマルテンサイト相と高い剛性のオーステナイト相とを、熱的にスイッチされる。マルテンサイト相におけるアクチュエータの形状はオーステナイトの形状に関係して変形される。その形状の変化が滴の射出を引き起こす。	強い力 (数百 MPa の応力) が得られる。 大きな歪み (3 % 以上) が得られる。 高い耐食性。 製造が簡単。 シングルノズルからページ幅ブリントヘッドへの拡張は容易。 低電圧駆動	最大数のサイクルの疲労限界。疲労抵抗を増加させるために小さな歪み (1 %) が必要。 熱除去のため、サイクル速度には限界がある。 一般的でない材料 (TiNi) が必要である。 変質に伴う潜熱が生じる。 大電流駆動。 マルテンサイト状態にするためにプレストレスが必要。	IJ26
リニア電磁アクチュエータ	リニア電磁アクチュエータとしては、リニア誘導アクチュエータ (LIA) や、リニア永久磁石同期アクチュエータ (LPSA) や、リニア・リラクタンس同期アクチュエータ (LRSA) や、リニア・スイッチド・リラクタンス・アクチュエータ (LSRA) や、リニア・ステップ・アクチュエータ (LSA) を挙げることができる。	リニア電磁アクチュエータは、ブレーナ半導体製造技術を使用し、高い信頼性、長い移動、高い効率を達成することができる。 アクチュエータの長い移動距離が可能となる。 中位の力が得られる。 低電圧駆動	軟磁性合金 (例えば、CoNiFe [1]) のような、一般的で無い半導体材料が必要である。 幾つかの種類は、ネオジム・鉄・ボロン系 (NdFeB) のような永久磁石材料が必要である。 複雑な多層の駆動回路が必要である。 大電流駆動。	IJ12

10

【1 8 9 2】

【表 2 2 1】

基本オペレーションモード

オペレーションモード	説明	メリット	デメリット	例
アクチュエータが直接インクを押し出す。	これは最もシンプルなオペレーションモードである。すなわち、アクチュエータは滴を押し出すために十分な運動エネルギーを直接供給する。滴は、その表面張力に打ち勝つための十分な速度を有する。	操作が簡単。 外部場は不要 滴の速度を 4 m/s より遅くすれば、付随滴 (の発生) を避けることができる。 使用されるアクチュエータに応じて、効率化できる。	滴の反復速度は、通常は 10 KHz 以下に制限される。しかしながら、これはその方法の基本ではなく、通常に使用される補給方法に関連するものである。 滴の全運動エネルギーはアクチュエータにより与えられる。 滴の速度が 4, 5 m/s よりも速ければ、通常は付随液滴が形成される。	サーマルインクジェット ピエゾインクジェット IJ01, IJ02, IJ03, IJ04 IJ05, IJ06, IJ07, IJ09 IJ11, IJ12, IJ14, IJ16 IJ20, IJ22, IJ23, IJ24 IJ25, IJ26, IJ27, IJ28 IJ29, IJ30, IJ31, IJ32 IJ33, IJ34, IJ35, IJ36 IJ37, IJ38, IJ39, IJ40 IJ41, IJ42, IJ43, IJ44
近接	プリントされる滴は幾つかの方法 (例えば、加圧インクにおける、熱的に誘導される表面張力減少) によって選ばれる。選択された滴は、プリントメディアや転写ローラーとの接触によってノズル中のインクから分離される。	とても簡単なプリントヘッド製造 (方法) を使用することができる。 その滴の選択方法は、ノズルから滴を分離するためのエネルギーを必要としない。	プリントヘッドとプリントメディアや転写ローラーとが近いことが要求される。 画像列を交互にプリントするため、2つのプリントヘッドが必要となるかも知れない。 モノリシック構造のカラープリントヘッドが困難である。	シルバーブルック、EP 0771 658 A2、及び関連特許出願
静電気でインクを引き付ける。	プリントされる滴は幾つかの方法 (例えば、加圧インクにおける、熱的に誘導される表面張力減少) によって選ばれる。選択された滴は、強い電場によってノズル中のインクから分離される。	とても簡単なプリントヘッド製造 (方法) を使用することができる。 その滴の選択方法は、ノズルから滴を分離するためのエネルギーを必要としない。	非常に高い静電場が必要である。 小さなノズルサイズのための静電場は空気の絶縁破壊以上となる。 静電場は埃を引き付けるかも知れない。	シルバーブルック、EP 0771 658 A2、及び関連特許出願 Tone-jet

20

30

【1 8 9 3】

【表 2 2 2】

磁気でインクを引き付ける。	プリントされる滴は幾つかの方法（例えば、加圧インクにおける、熱的に誘導される表面張力減少）によって選ばれる。選択された滴は、磁気インクに作用する強い磁場によってノズル中のインクから分離される。	とても簡単なプリントヘッド製造（方法）を使用することができる。その滴の選択方法は、ノズルから滴を分離するためのエネルギーを必要としない。	磁気インクが必要となる。黒以外の色のインクが難しい。非常に高い磁場が必要である。	シルバーブルック、EP 0771 658 A2、及び関連特許出願
シャッター	アクチュエータは、インクがノズルに流れるのを阻止するためにシャッターを動かす。インク圧は、多様な滴の射出周波数に脈動している。	補給時間の削減のため、50 KHzより高速の動作が可能である。滴のタイミングを非常に正確にできる。アクチュエータのエネルギーは極めて低くできる。	可動部が必要である。インク圧モジュレータが必要である。摩擦と摩耗を考慮しなければならない。静止摩擦力が必要。	IJ13, IJ17, IJ21
シャッターグリル	アクチュエータは、インクがグリルを経てノズルに流れるのを阻止するためにシャッターを動かす。そのシャッターの移動はグリルホルルの幅に等しいだけ行う必要がある。	小規模のアクチュエータを使用できる。小さな力のアクチュエータでも使用できる。50 KHzより高速の動作が可能である。	可動部が必要である。インク圧モジュレータが必要である。摩擦と摩耗を考慮しなければならない。静止摩擦力が必要。	IJ08, IJ15, IJ18, IJ19
磁気パルスがインク・ブッシャーを引き付ける。	パルス化された磁場が、滴射出周波数で「インク・ブッシャー」を引き付ける。アクチュエータは、滴が射出されるべきで無い時にインク・ブッシャーの移動を阻止するためのキャッチを制御する。	極めて低いエネルギーでの操作が可能である。熱損失の問題は無い。	外部のパルス磁場が必要である。アクチュエータとインクブッシャーの両方に特殊な材料が必要である。製造が複雑	IJ10

10

【1 8 9 4】

【表 2 2 3】

20

補助装置のメカニズム（全てのノズルに適用される。）

補助装置のメカニズム	説明	メリット	デメリット	例
なし	アクチュエータは直接インク滴を射出する。そして、外部場や他のメカニズムは必要とされない。	製造が簡単。 操作が簡単。 物理的寸法が小さい。	滴の射出エネルギーは個々のノズルアクチュエータから供給されなければならない。	ピエゾエレクトリックやサーマルバルブを含む、ほとんどのインクジェット IJ01-IJ07, IJ09, IJ11 IJ12, IJ14, IJ20, IJ22 IJ23-IJ45
インク圧を変動させる。（音響シミュレーションを含む）	インク圧を変動させ、多くの滴射出エネルギーを生じさせる。アクチュエータは、ノズルを選択的にブロックしたり有効にしたりして、射出されるべき滴を選択する。インク圧の変動は、プリントヘッドを振動させることによって成し遂げられ、又は、好ましくは、インクの中のアクチュエータによって成し遂げられるかも知れない。	インク圧の変動は補給パルスを生じさせ、高い駆動スピードが得られる。アクチュエータはかなり低いエネルギーで駆動される。音響レンズがノズルに音波を収束させるために使用できる。	外付けのインク圧発生器が必要である。インク圧の位相や振幅は注意深く制御されなければならない。インクチャンバー内における音響反射は設計されなければならない。	シルバーブルック、EP0771 658 A2、及び関連特許出願 IJ08, IJ13, IJ15, IJ17 IJ18, IJ19, IJ21
メディアの接近	プリントヘッドはプリントメディアに近接するように配置される。選択されていない滴ではなく、選択された滴がプリントヘッドから押し出され、プリントメディアに接触する。滴は、滴の分離が生じない程に早く、プリントメディアに浸透する。	低電力 高精度 プリントヘッドの構造が簡単。	精密な組み立てが必要である。紙の繊維が問題を引き起こすかも知れない。でこぼこのある面にはプリントできない。	シルバーブルック、EP0771 658 A2、及び関連特許出願
転写ローラー	滴は、プリントメディアに直接ではなく、一旦転写ローラーに付着する。転写ローラーは近接した滴を分離するためにも使用される。	高精度 広い幅の被印刷物を使用できる。 転写ローラー表面に付着したローラーは乾く。	大きい 高価 製造が複雑	シルバーブルック、EP0771 658 A2、及び関連特許出願 テクトロニクス・ホットメルト・ピエゾエレクトリック・インクジェット 如何なるIJシリーズ

30

40

【1 8 9 5】

【表 2 2 4】

静電気	選択された滴をプリントメディアの方へ加速させるために静電場が使用される。	低電力 プリントヘッドの構造が簡単。	小滴の分離に必要な電場強度は、空気の絶縁破壊の近傍かそれ以上である。	シルバーブルック, EP0771 658 A2、及び関連特許出願 Tone-jet
直流磁場	磁場が、磁気インクの選択された滴をプリントメディアの方向へ加速させるために使用される。	低電力 プリントヘッドの製造が簡単。	磁気インクが必要となる。強い磁場が必要。	シルバーブルック, EP0771 658 A2、及び関連特許出願
交流磁場	プリントヘッドは一定の磁場の中に置かれる。通電ワイヤの中のローレンツ力はアクチュエータを動かすために使用される。	プリントヘッド製造工程において、統合する (integrate) ための磁性材料は必要とされない。	外部磁界が必要である。エレクトロマイグレーション問題の結果、電流密度は高くなるかも知れない。	IJ06, IJ16
パルス化された磁場	パルス化された磁場は周期的にパドルを引き付けるために使用され、パドルはインクを押し出す。パドルの移動を選択的に阻止するためのキャッチが、アクチュエータにより駆動される。	非常に低い電力による駆動が可能である。 プリントヘッドのサイズが小さい。	プリントヘッドの製造が複雑。プリントヘッドに磁性材料が必要。	IJ10

10

【1 8 9 6】

【表 2 2 5】

アクチュエータの増幅、或いは変更方法

アクチュエータの増幅	説明	メリット	デメリット	例
なし	アクチュエータの機械的増幅は使用されない。アクチュエータは滴射出工程を直接的に遂行する。	操作簡単	多くのアクチュエータメカニズムは、滴射出工程を効率的に促進するには、不十分な移動距離と、不十分な力である。	サーマルバルブインクジェット IJ01, IJ02, IJ06, IJ07 IJ16, IJ25, IJ26
異なる膨張 曲げアクチュエータ	アクチュエータの材料は他側に比べて一側の方が膨張する。その膨張は、熱や、ピエゾエレクトリックや、磁気歪や、他のメカニズムによるものである。	縮小されたプリントヘッドエリアにおいて、より長い移動距離が得られる。 前記曲げアクチュエータは、大きな力で移動距離の小さいアクチュエータメカニズムを、移動距離が大きく力の小さいメカニズムに変換する。	歪みが大きい。 材料が薄い層に裂けないことを考慮されなければならない。 形成される間、高温や高ストレスに起因する残留曲げ。	ピエゾエレクトリック IJ03, IJ09, IJ17-IJ24 IJ27, IJ29-IJ39, IJ42, IJ43, IJ44
過渡曲げアクチュエータ	外側の2つの層が全く同一である、3層の曲げアクチュエータ。これは、大気温度や残留ストレスによる曲げを解消させる。そのアクチュエータは、一側又は他側に熱を加えたときにのみ反応する。	温度安定性が大変に良い。 新しい液滴は熱が無くなる前に射出されるので、高速である。 残留応力が除去される。	歪みが大きい。 材料が薄い層に裂けないことを考慮されなければならない。	IJ40, IJ41
アクチュエータの積み重ね	一連の薄いアクチュエータが積み重ねられる。 このことは、静電気、及びピエゾエレクトリックアクチュエータのように、アクチュエータが高電界強度を必要とする場合に適している。	移動距離が増加。 駆動電圧が低い。	製造の複雑性が増す。 ピンホールに伴う短絡のおそれが増える。	幾つかのピエゾエレクトリックインクジェット IJ04
多数のアクチュエータ	多数の小さなアクチュエータがインクを吐出するのに同時に使用される。各アクチュエータは、所望の力の一部分のみを提供するためにだけ必要である。	アクチュエータから得られる力が大きい。 インク流を正確にコントロールするために、多数のアクチュエータを配置できる。	アクチュエータの力は線形的には増えず、効率が低下する。	IJ12, IJ13, IJ18, IJ20 IJ22, IJ28, IJ42, IJ43

20

30

【1 8 9 7】

【表 2 2 6】

線形バネ	線形バネが、小さな移動距離で大きな力の動きを、より長い移動距離でより小さい力の動きに変換するのに使用される。	短い移動距離のアクチュエータでありながら、より長い移動距離の要求に合致させる。移動を変換する非接触の方法。	バネを配置する分だけの面積が必要。	IJ15
逆転バネ	アクチュエータはバネに負荷をかける。アクチュエータがオフにされると、バネは元に戻る。これにより、アクチュエータの力／移動距離曲線を逆にでき、滴射出の力／移動距離の要求に適合させることができる。	インクにより適合する。	製造が複雑。 スプリングに大きな応力	IJ05, IJ11
コイル状のアクチュエータ	曲げアクチュエータは狭いチップ面積中で長い移動を可能とすべく、グルグル巻かれている。	移動距離が増加。 チップ面積の減少 ブレーナの実装が、製造するのに比較的簡単である。	他の方向での製造が困難であるため、ブレーナの実装が一般には制限される。	IJ17, IJ21, IJ34, IJ35
屈曲曲げアクチュエータ	曲げアクチュエータは、屈曲ポイントの近くに小さな領域を有する。その屈曲ポイントは、残ったアクチュエータよりもより迅速に曲げる。そのアクチュエータの屈曲は、緩やかなコイルが角張った曲げに効率よく変換され、アクチュエータのより長い移動距離を達成できる。	曲げアクチュエータの移動距離を増加させるための簡単な装置である。	屈曲部分が弾性限界を超えないように注意しなければならない。 ストレスは非常に平坦に分布する。 有限要素解析でモデルを正確に解析するのは困難である。	IJ10, IJ19, IJ33
ギヤ	ギヤは、期間を犠牲にして、移動距離を増すために使用される。円形ギヤや、ラックアンドピニオンや、ラチェットや、他のギヤが使用される。	小さな力で小さな移動距離のアクチュエータを使用することができる。 標準的な表面MEMS工程を使用して製造することができる。	可動部分が必要。 幾つかのアクチュエータサイクルが必要。 電子機器を駆動するのが複雑である。 製造に手間がかかる。 摩擦や摩耗が生じる。	IJ13

10

20

【1 8 9 8】

【表 2 2 7】

キャッチ	アクチュエータは小さなキャッチをコントロールする。そのキャッチは、大部分の方法でコントロールされる、インク・プッシャーの稼働を有効或いは無効にする。	アクチュエータのエネルギーが非常に小さい。 アクチュエータのサイズが非常に小さい。	製造に手間がかかる。 外力が必要。 顔料インクに適していない。	IJ10
バックルプレート	バックルプレートは、ゆっくりとしたアクチュエータを速いアクチュエータに変化させるために使用される。それは、また、強い力で短い移動距離のアクチュエータを、長い移動距離で中程度の力の動きに変換することができる。	非常に速い動きを実現できる。	装置寿命を長くするために材料の弾性限度内に保持しなければならない。 大きなストレスが生じる。 一般に高出力が要求される。	平田ら、“インクジェットヘッド ... ”, Proc. IEEE MEMS, Feb. 1996, pp418-423 IJ18, IJ27
先細型の磁石ポール	先細型の磁石ポールは、力を犠牲にして移動距離を増すことができる。	磁力／距離曲線が線形化される。	製造に手間がかかる。	IJ14
レバー	レバーと支点が、小さな移動距離で大きな力の動きを、より長い移動距離で比較的小さな力の動きに変換する。そのレバーはまた、移動の方向を逆向きにすることができる。	短い移動距離のアクチュエータでありながら、より長い移動距離の要求に合致させる。 支点のエリアは線形の動きをしない。流体シールを使用できる。	支点の周囲が高応力である。	IJ32, IJ36, IJ37
ロータリー・インペラ	アクチュエータはロータリー・インペラに接続されている。そのアクチュエータの小さな角度の偏差は、インペラの羽根の回転を生じさせる。該羽根は、インクを押し、ノズルから吐出させる。	大きな機械的なメリット。 アクチュエータの、移動距離に対する力の比は、インペラの羽根の数を変化させることにより、ノズルの要求に適合させることができる。	製造に手間がかかる。 顔料インクには適さない。	IJ28
音響レンズ	屈折又は回折（例えば、ゾーンプレート）の音響レンズは音波を集中させるために使用される。	可動部なし。	大きな面積が必要。 音響インクジェットのみに関係する。	1993年、Hadimioglu ら、EUP550, 192 1993年、Elrod ら、EUP572, 220

30

40

【1 8 9 9】

【表 2 2 8】

尖った、導電性の点	静電場を集中させるために鋭く尖った部分が使用される。	製造が簡単。	表面吐出インクジェットのために標準的なVLSIプロセスを使用して製造することは困難である。 静電インクジェットにのみ関係する。	Tone-jet

50

【1900】

【表229】

アクチュエータの動き

アクチュエータの動き	説明	メリット	デメリット	例
体積膨張	四方八方にインクを押すため、アクチュエータの体積が変化する。	サーマルインクジェットの中では構造が簡単。	大体は、体積膨張のために大きなエネルギーが必要とされる。これにより、サーマルインクジェットの実施に当たり、熱応力や、キャビテーションや、kogationが発生する。	ヒューレットパカード・サーマルインクジェット キャノン バブルジェット
線状、チップ表面の法線	アクチュエータはプリントヘッド表面の法線方向へ移動する。ノズルは大体は移動中である。	表面の法線に射出されたインク滴への効率的な結合。	垂直方向の動きを成し遂げるには、製造が複雑となる。	IJ01, IJ02, IJ04, IJ07 IJ11, IJ14
線状、チップ表面に平行	アクチュエータは、プリントヘッド表面に沿って平行に動く。滴の噴出は表面の法線方向である。	プレーナーの形成に向いている。	製造が複雑。 摩擦 静止摩擦力	IJ12, IJ13, IJ15, IJ33 IJ34, IJ35, IJ36
薄膜押し	強い力でありながら小型のアクチュエータが、インクと接触している丈夫な膜を押すために使用される。	アクチュエータの有効面積が膜面積になる。	製造が複雑。 アクチュエータサイズ VLSI プロセスへの統合が困難	1982, ホーキンス、USP 4, 459, 601
回転	アクチュエータは、グリルやインペラのような幾つかの要素の回転をもたらす。	道程を増加させるためにロータリーレバーが使用されるかも知れない。 チップ面積は小さくて良い。	デバイスが複雑 回転中心にて摩擦を生じるかも知れない。	IJ05, IJ08, IJ13, IJ28
曲げ	アクチュエータは電圧印加により曲がる。これは、異なる熱膨張や、ピエゾエレクトリックの膨張や、磁気歪や、他の相対的な寸法変化の方法による。	非常に小さな寸法変化を大きな動きに変換させることができる。	少なくとも2つのはっきりと区別できる層からアクチュエータが形成されることが必要である。或いは、アクチュエータに沿って、熱的な差を有するようにすることが必要である。	1970年、カイザーら、 USP3, 946, 398 1973年、Stemme、 USP3, 747, 120 IJ03, IJ09, IJ10, IJ19 IJ23, IJ24, IJ25, IJ29 IJ30, IJ31, IJ33, IJ34 IJ35

10

20

【1901】

【表230】

旋回	アクチュエータは中心枢軸の回りを旋回する。この動きは、パドルの反対側に力（例えば、ローレンツ力）が付与された場合に適している。	パドルへの線形の力が零となった場合に操作が許容される。 チップ面積は小さくて良い。	インクの動きへの非効率な結合。	IJ06
真っ直ぐ	アクチュエータは、普段は曲がっていて、電圧印加時に真っ直ぐになる。	オーステナイト相がプレーナである形状記憶合金に使用できる。	静止曲げが正確であることを裏付けるため、ストレスの注意深いバランスが必要とされる。	IJ26, IJ32
二重曲げ	一つの要素に電圧が印加されたときにアクチュエータは1方向に曲がり、もう一つの要素に電圧が印加されたときには該アクチュエータは他の方向に曲がる。	1つのアクチュエータが2つのノズルにパワーを供給するために使用される。 チップサイズは小さい。 環境温度の影響を受けにくい。	2つの曲げ方向に同じように滴を射出することは困難。 同程度のシングル曲げアクチュエータと比較して、小さな効率のロスがある。	IJ36, IJ37, IJ38
シエア	アクチュエータへの電圧印加がアクチュエータ材料におけるシエア・モーションを引き起こす。	ピエゾエレクトリック・アクチュエータの有効な移動距離を増加することができる。	他のアクチュエータ装置に直ちに適用させることはできない。	1985年、フィッシュベック、 USP4, 584, 590
放射状の収縮	アクチュエータはインクリザーバを強く締め付ける。インクは、締め付けられたノズルから押し出される。	巨視的な構造であるガラスチューブからシングルノズルを製造することは、比較的簡単である。	大きな力が要求される。 非効率的 VLSI プロセスで集積することが困難。	1970, Zoltan USP3, 683, 212
グルグル巻く／真っ直ぐに伸ばす	巻かれたアクチュエータは、真っ直ぐに伸びるか、よりきつく巻かれる。そのアクチュエータの自由端の動きがインクを射出する。	プレーナVLSIプロセスでの製造が簡単 面積が小さくて済み、コストを低くできる。	ノンプレーナデバイスの製造が困難。 面外剛性に乏しい。	IJ17, IJ21, IJ34, IJ35
弓	電圧が印加されたとき、アクチュエータは真ん中が曲がる（又は折れる）。	移動スピードを速めることができる。 剛体である。	長い移動は制約を受ける。 大きな力が要求される。	IJ16, IJ18, IJ27
押す－引く	2つのアクチュエータがシャッターをコントロールする。1つはシャッターを引き、もう一方はそれを押す。	構造体は両端が固定されているので、高い面外剛性を有する。	直接インクを押し出すインクジェットに、容易には適さない。	IJ18

30

40

【1902】

【表 2 3 1】

内側にカールする。	一組のアクチュエータが、内包するインクの体積を減少させるように、内側にカールする。	アクチュエータの背後部分への良好な流体の流れが、効率を増す。	デザインが複雑。	IJ20, IJ42
外側にカールする。	一組のアクチュエータが外側にカールし、アクチュエータの周りのチャンバーにあるインクに圧力を加え、チャンバーのノズルからインクを吐出する。	比較的製造が簡単。	チップ面積は比較的大きい。	IJ43
絞りのように閉まる。	多数の翼が多量のインクを封入する。これらはいっせいに回転し、翼間の体積を減少させる。	高効率 チップ面積が小さい。	製造がかなり複雑。 顔料インクには適さない。	IJ22
音響振動	アクチュエータは高周波数で振動する。	アクチュエータはインクから離すことができる。	使用周波数にて効率的な運転を行うには大きな面積が必要とされる。 音響結合とクロストーク。 駆動回路が複雑である。 滴の大きさや位置をコントロールしにくい。	1993年、Hadimioglu ら、EUP550, 192 1993年、Elrod ら、EUP572, 220
無し	様々なインクジェット設計において、アクチュエータは動かない。	可動部無し。	可動部を撤廃するため、他の様々なトレードオフが必要とされる。	シルバールック, EP0771 658 A2、及び関連特許出願 Tone-jet

10

【1903】

【表 2 3 2】

ノズル補給方法

ノズル補給方法	説明	メリット	デメリット	例
表面張力	アクチュエータは動作した後、その標準位置に素早く戻る。その素早い戻りは、ノズル開口から空気を吸収する。そして、ノズルでのインクの表面張力は、メニスカスを最小面積に戻す小さな力を与える。	製造が簡単 操作が簡単	低速 表面張力はアクチュエータの力と比較して相対的に小さい。 長い補給時間は、通常は、合計反復速度を左右する。	サーマルインクジェット ピエゾエレクトリックインクジェット IJ01-IJ07, IJ10-IJ14 IJ16, IJ20, IJ22-IJ45
シャッターされ、発振されるインク圧力	ノズルチャンバーへのインクは、滴射出周波数の2倍で発射される圧力によって補給される。滴が射出される時、シャッターは3半サイクル（すなわち、滴の射出、アクチュエータの戻り、補給）だけ開けられる。	高速 アクチュエータは、インク滴を射出するのではなくシャッターを開閉するだけで良く、低エネルギーで良い。	普通のインク圧力の発振が必要。 顔料インクには適していないかも知れない。	IJ08, IJ13, IJ15, IJ17 IJ18, IJ19, IJ21
補充のアクチュエータ	メインアクチュエータが滴を射出した後、第2の（補充）アクチュエータには電圧が印加される。該補充アクチュエータはノズルチャンバーにインクを注入する。再びチャンバーが空にならないようにするために、その補充アクチュエータはゆっくりと戻る。	ノズルが活発に補充されるので、高速。	ノズル毎に2つの独立したアクチュエータが必要。	IJ09
正のインク圧	インクは、僅かな正圧に維持される。インク滴が吐出された後は、表面張力とインク圧力の両方がノズルを満たすように作用するので、ノズルチャンバーは素早く充填される。	高い補充速度、したがって、高い滴の吐出速度が可能となる。	表面からの溢れ出しを防がなければならない。 プリントヘッド表面が疎水性に富むことが要求される。	シルバールック, EP0771 658 A2、及び関連特許出願 代替： IJ01-IJ07, IJ10-IJ14 IJ16, IJ20, IJ22-IJ45

20

30

【1904】

【表 2 3 3】

注入口での逆流を減少させる方法

注入口での逆流を減少させる方法	説明	メリット	デメリット	例
長い注入通路	ノズルチャンバーへのインク注入通路を長く、そして相対的に狭くする。注入口での逆流を減少させるために粘性抵抗を当てにする。	設計が簡単 操作が簡単 クロストークが減少	補充速度は遅い。 相対的に大きなチップ面積になるかも知れない。 部分的な効果のみ。	サーマルインクジェット ピエゾエレクトリックインクジェット IJ42, IJ43
正のインク圧	インクは正圧なので、静止状態では、幾つかのインク滴は既にノズルから飛び出る。これにより、一定の体積のインクを吐出させることが要求されるノズルチャンバーの圧力が低下する。このチャンバー圧力の低下は、注入口でのインクの押し出しを減少させる。	滴の選択や分離のための力を減少させることができる。 補給時間が短い。	プリントヘッドの吐出面の溢れ出しを防止する方法（ノズルリムや、効果的な疎水処理や、それらの両方のような方法）が必要である。	シルバールック、EP0771 658 A2、及び関連特許出願 次の駆動が可能。すなわち、 IJ01-IJ07, IJ09-IJ12 IJ14, IJ16, IJ20, IJ22 IJ23-IJ34, IJ36-IJ41 IJ44
バッフル	1又はそれ以上のバッフルが注入口のインク流中に配置される。アクチュエータに電圧が印加されたとき、注入口を通り抜ける流れを制限する滴が、速いインクの移動によって発生される。ゆっくりとした補給の工程は制限されず、渦も生じない。	補充速度は、長い注入通路法と同じようには遅くはならない。 クロストークが減少	デザインが複雑。 製造の複雑さが増加するかも知れない（例えば、テクトロンクス・ホットメルト・ピエゾエレクトリック・プリントヘッド）	HPサーマルインクジェット テクトロンクス・ピエゾエレクトリック・インクジェット プリントヘッド
フレキシブルなフラップが注入口を制限する。	最近キヤノンにより開示されたこの方法では、膨張アクチュエータ（気泡）がフレキシブルなフラップを押し、注入口を制限する。	エッジシューターのサーマルインクジェットデバイスにとって、逆流がかなり減少される。	ほとんどのインクジェット構造に適用できない。 製造が複雑になる。 ポリマーフラップの非弾性変形が、使用限界のクリープを生じさせる。	キヤノン

10

20

【1905】

【表 2 3 4】

注入口のフィルタ	インク注入口とノズルチャンバーとの間にフィルタが配置される。そのフィルタは、インク流を制限する、多数の小さな孔やスロット（細長い孔）を有している。そのフィルタはまた、ノズルを詰まらせるかも知れない粒子を取り除く。	インクの濾過を付加できるメリット。 インクフィルタが、工程を付加することなく製造されるかも知れない。	補充速度は遅い。 構造が複雑になるかも知れない。	IJ04, IJ12, IJ24, IJ27 IJ29, IJ30
ノズルと比較して小さな注入口	ノズルチャンバーに繋がるインク注入路の断面積は、ノズルのものより大幅に小さい。したがって、注入口からよりもノズルからインクが出やすい。	デザインが簡単。	補充速度は遅い。 相対的に大きなチップ面積になるかも知れない。 部分的な効果のみ。	IJ02, IJ37, IJ44
注入口のシャッター	第2のアクチュエータはシャッターの位置をコントロールする。メインアクチュエータに電圧が印加されると、インクの注入口が閉塞される。	インクジェットプリントヘッドの動作スピードが増加する。	補充アクチュエータと駆動回路とを分離させることが必要。	IJ09
注入口は、インク押し出し面の後ろに配置される。	その方法は、アクチュエータのインク押し出し面を注入口とノズルとの間に配置することにより、注入口におけるインクの逆流という問題を避ける。	インクの逆流問題は解消されている。	パドルの背面の負圧を抑えるため、注意深い設計が要求される。	IJ01, IJ03, IJ05, IJ06 IJ07, IJ10, IJ11, IJ14 IJ16, IJ22, IJ23, IJ25 IJ28, IJ31, IJ32, IJ33 IJ34, IJ35, IJ36, IJ39 IJ40, IJ41
アクチュエータの一部が注入口を閉じるために動く。	アクチュエータ、及びインクチャンバーの壁がアレンジされ、アクチュエータが吸入口を閉塞するように動く。	逆流をかなり減少させることができる。 コンパクトなデザインが可能。	製造の複雑さが微増する。	IJ07, IJ20, IJ26, IJ38
ノズルアクチュエータはインクの逆流を引き起こさない。	インクジェットの幾つかの構成においては、注入口でのインクの逆流を引き起こしてしまうような、アクチュエータの膨張や移動は無い。	インクの逆流問題は解消されている。	作動に、インクの逆流は関係しない。	シルバールック、EP0771 658 A2、及び関連特許出願 バルブジェット Tone-jet IJ08, IJ13, IJ15, IJ17 IJ18, IJ19, IJ21

30

40

【1906】

【表 2 3 5】

ノズルをきれいにする方法	説明	メリット	デメリット	例
通常のノズルの射出	全てのノズルは、乾く前にインクを定期的に射出する。使用されない場合、ノズルは空気からシール（キャップ）される。そのノズルの射出は、通常は、プリントヘッドがクリーニングステーションに最初に移動してきた後、特別の掃除サイクルの間になされる。	プリントヘッドに、複雑な構成は付加されない。	乾いたインクを除去するのには適さない。	ほとんどのインクジェットシステム IJ01-IJ07, IJ09-IJ12 IJ14, IJ16, IJ20, IJ22 IJ23-IJ34, IJ36-IJ45
インクヒーターへの特別のパワー	通常の状態ではインクを沸騰させるのではなく暖めるシステムにおいては、ノズルの掃除は、ヒータに過剰な電力を供給してインクをノズルにて沸騰させることにより達成される。	もし、ヒーターがノズルに近接していたら、非常に効率的になる。	掃除のために、より高い駆動電圧が必要となる。より大きな駆動トランジスタが必要となるかも知れない。	シルバールック, EP0771 658 A2、及び関連特許出願
連続的なアクチュエータへのパルス	アクチュエータは連続的に駆動される。幾つかの機器構成においては、このことが、ノズルにおいてインクが沸騰されるという熱集積を引き起こし、ノズルが掃除される。他の状態では、ノズルの詰まりを除去するための十分な振動を引き起こすかも知れない。	プリントヘッドには特別な駆動回路は必要とされない。デジタル論理によりコントロールされ、初期化することができる。	インクジェットノズルの構造に大いに影響される。	以下で使用されるかも知れない。 IJ01-IJ07, IJ09-IJ11 IJ14, IJ16, IJ20, IJ22 IJ23-IJ25, IJ27-IJ34 IJ36-IJ45

10

【1 9 0 7】

【表 2 3 6】

20

インク押し出し用アクチュエータへの特別のパワー	通常、アクチュエータが動作限界にまで駆動されていない場合、そのアクチュエータに増強した駆動信号を与えることによってノズル掃除が行われるかも知れない。	適用できるなら、ノズル掃除方法としては簡単な方法である。	アクチュエータの動きに強い限界がある場合には適さない。	以下で使用されるかも知れない。 IJ03, IJ09, IJ16, IJ20 IJ23, IJ24, IJ25, IJ27 IJ29, IJ30, IJ31, IJ32 IJ39, IJ40, IJ41, IJ42 IJ43, IJ44, IJ45
音響共振	インクチャンバーに超音波が供給される。この超音波は、詰まりを掃除するための十分な力をノズルに生じさせるために、適当な振幅と周波数とを有する。もし、供給される超音波の周波数が、インクの穴（cavity）の共鳴周波数と等しければ、実現が容易である。	ノズル掃除能力が高い。既に音響アクチュエータを有するシステムにおいては非常に低いコストで実装されるかも知れない。	システムが音響アクチュエータを有していなければ、実装のための費用は高くなる。	IJ08, IJ13, IJ15, IJ17 IJ18, IJ19, IJ21
ノズル掃除プレート	微細加工したプレートがノズルに押し付けられる。そのプレートは全てのノズルのために柱を有している。柱の列	ひどく詰まったノズルを掃除できる。	正確な機械の配列が必要である。可動部が必要である。ノズルが損傷するおそれがある。精密な製造が要求される。	シルバールック, EP0771 658 A2、及び関連特許出願
インク圧力パルス	インクが全ノズルから噴き出すように、インク圧は一時的に増加される。これは、アクチュエータへの電圧印加と同時に使用されるかも知れない。	他の方法を使用できない所に効果が出るかも知れない。	圧力ポンプや他の圧力機器が必要とされる。高価 インクが無駄。	全ての IJ シリーズのインクジェットで使用されるかも知れない。
プリントヘッドワイパー	フレキシブルな「ブレード」がプリントヘッド表面を横切りながら拭く。そのブレードは、一般的には、ゴムや剛性高分子弾性体のようなフレキシブルポリマーから製造される。	平坦なプリントヘッド表面に効果がある。 安価	平面でない、或いは非常に脆弱なプリントヘッド表面への使用は困難である。機械部品が必要である。プリントシステムが大きいためにブレードが摩耗してしまう。	多くのインクジェットシステム

30

【1 9 0 8】

【表 2 3 7】

40

分離された、インクを沸騰させるヒータ	通常の滴射出装置はそれを要求していないけれども、分離されたヒーターがノズルに配置されている。多くのノズルが同時に掃除されるので、それらのヒーターは個々の駆動回路を必要とせず、画像化も必要とされない。	他のノズル掃除方法を使用できないような箇所に効果がある。幾つかのインクジェット構造においては、追加費用無しに実装することができる。	製造が複雑。	多くの IJ シリーズのインクジェットで使用するができる。

【1 9 0 9】

【表 2 3 8】

ノズルプレート構造

ノズルプレート構造	説明	メリット	デメリット	例
電気鋳造されたニッケル	ノズルプレートが、電気鋳造されたニッケルから分離するように製造され、プリントヘッドチップに結合される。	製造が簡単。	高温と高圧がノズルプレートを結合するために必要である。最小限の薄さが矯正される。異なる熱膨張。	ヒューレットパッカードサーマルインクジェット
レーザーによる剥ぎ取り、又は穴開けされたポリマー	ノズルプレートに強烈な UV レーザにより個々のノズル孔が形成される。ノズルプレートは、一般的には、ポリイミドやポリスルホンのようなポリマーである。	マスクは必要としない。かなり速くなる。ノズルの特徴を超えた何らかのコントロールが可能となる。必要な装置は比較的安価である。	各穴を個別に形成しなければならない。特別な装置が必要となる。プリントヘッドに数千のノズルを有する場合は遅い。排出口に薄いバリが出来るかも知れない。	キヤノン パブルジェット 1988 Sercel ら、SPIE、Vol. 998 Excimer Bean Applications, pp. 76-83 1993、渡辺ら、USP5, 208, 604
マイクロ機械加工されたシリコン	単結晶シリコンから分離されたノズルプレートがマイクロ機械加工され、プリントヘッドのウエハに結合される。	高精度を達成できる。	2つの部分の構造 高価 精密な位置合わせが必要となる。接着剤によりノズルが詰まるかも知れない。	K. Bean, IEEE 電子デバイスの取扱、Vol. ED-25, No. 10, 1978, pp1185-1195 ゼロックス、1990年、1990、ホーキンスら、USP 4, 899, 181
ガラス毛管	細いガラス毛管がガラスチューブから引き抜かれる。この方法は個々のノズルを作るために使用されてきたが、数千ものノズルを有するプリントヘッドを大量製造するのに使用するのには困難である。	高価な装置は不要。シングルノズルとするのが簡単。	非常に小さなノズルを形成するのは困難である。大量生産に適していない。	1970, Zoltan USP3, 683, 212

10

【1910】

【表 2 3 9】

モノリシック、VLSI リソグラフィ法を使ってマイクロマシン加工された表面	ノズルプレートは、標準的な VLSI 蒸着法を使って層として蒸着される。ノズルは、VLSI リソグラフィやエッチングを使ってノズルプレート中に形成される。	高精度 (<1 μm) モノリシック 安価 存在するプロセスを使用できる。	ノズルチャンバーを形成するため、前記ノズルプレートの下側に犠牲層が必要とされる。表面は触ると壊れやすい。	シルバールック、EP0771 658 A2、及び関連特許出願 IJ01, IJ02, IJ04, IJ11 IJ12, IJ17, IJ18, IJ20 IJ22, IJ24, IJ27, IJ28 IJ29, IJ30, IJ31, IJ32 IJ33, IJ34, IJ36, IJ37 IJ38, IJ39, IJ40, IJ41 IJ42, IJ43, IJ44
モノリシック、基板を貫通するように腐食された。	ノズルプレートは、ウエハの中に埋もれたエッチング停止層である。ノズルチャンバーはウエハの正面に形成され、ウエハは裏面から薄くされる。その後、前記エッチング停止層中にノズルがエッチングされる。	高精度 (<1 μm) モノリシック 安価 展開は難しい。	長いエッチング時間が必要である。サポートウエハが必要である。	IJ03, IJ05, IJ06, IJ07 IJ08, IJ09, IJ10, IJ13 IJ14, IJ15, IJ16, IJ19 IJ21, IJ23, IJ25, IJ26
ノズルプレートなし	ノズルの詰まり防止のため、ノズルを完全に撤廃するために様々な方法が試みられてきた。これらはサーマルバブルメカニズムや音響レンズメカニズムを有している。	ノズルが詰まることは無い。	滴の位置を正確にコントロールするのが困難。クロストーク問題。	リコー、1995年、関谷ら、USP5, 412, 413 1993年、Hadimioğluら、EUP550, 192 1993年、Elrodら、EUP572, 220
桶状容器	それぞれの滴の吐出装置は、權（パドル）が移動される桶状容器を有している。ノズルプレートは有していない。	製造の複雑さが低減される。モノリシック	滴を発射する方向は wicking に敏感である。	IJ35
個々のノズルの代わりのノズルスリット	スリットの形成（多くのアクチュエータを囲むような形成）に伴うノズル孔の廃止は、ノズルの詰まりを減少させる反面、インクの表面波に起因するクロストークを増加させる。	ノズルが詰まることは無い。	滴の位置を正確にコントロールするのは困難。クロストーク問題。	1989年、斉藤ら、USP4, 799, 068

20

30

【1911】

40

【表 2 4 0】

滴の噴出方向

噴出方向	説明	メリット	デメリット	例
エッジ (‘エッジシューター’)	インク流はチップの表面に沿い、そして、インク滴はそのチップのエッジから射出される。	製造簡単 シリコンエッチングは不要 基板を経由しての良好な放熱。 機械的に強い。 チップの取扱いの容易さ	ノズルはエッジに限定される。 高解像度の実現は困難である。 高速のカラープリントのためには、1色毎に1つのプリントヘッドが必要である。	キヤノン バブルジェット 1979 遠藤ら、英特許 2,007,162 ゼロックス、ヒーター・イン・ビット、1990、ホーキンスら、USP 4,899,181 Tone-jet
表面 (‘ルーフシューター’)	インク流はチップの表面に沿い、そして、インク滴はそのチップの表面から、そのチップの面の法線方向に射出される。	バルクシリコンエッチングは要求されない。 シリコンは効率的な放熱をできる。 機械的に強い。	インクの最高速はかなり遅い。	ヒューレットパッカード T1J, 1982 Vaught ら、USP4,490,728 IJ02, IJ11, IJ12, IJ20 IJ22
チップの通り抜け、前方 (‘アップシューター’)	インク流はチップを通り抜け、そして、インク滴はそのチップの前面から射出される。	インクの流れは速い。 ページ幅プリントに適している。 ノズルの高密度実装のために製造コストが低くなる。	バルクシリコンエッチングが必要。	シルバールック, EP0771 658 A2、及び関連特許出願 IJ04, IJ17, IJ18, IJ24 IJ27-IJ45
チップの通り抜け、逆 (‘ダウンシューター’)	インク流はチップを通り抜け、そして、インク滴はそのチップの背面から射出される。	インクの流れは速い。 ページ幅プリントに適している。 ノズルの高密度実装のために製造コストが低くなる。	ウェハの薄膜化が必要である。 製造中、特別の取扱いが必要とされる。	IJ01, IJ03, IJ05, IJ06 IJ07, IJ08, IJ09, IJ10 IJ13, IJ14, IJ15, IJ16 IJ19, IJ21, IJ23, IJ25 IJ26
アクチュエータの通り抜け	インク流はアクチュエータを通り抜ける。そのアクチュエータは、駆動用トランジスタと同じ基板には形成されていない。	ピエゾエレクトリックプリントヘッドに適している。	ページ幅プリントヘッドは駆動回路への数千の接続を必要とする。 一般的な CMOS 製造方法では製造できない。 複雑な組み立てが必要である。	エプソンスタイラス テクトロニクスホットメル ・ピエゾエレクトリック・インクジェット

10

【1 9 1 2】

【表 2 4 1】

20

インクタイプ	説明	メリット	デメリット	例
水性、染料	水と染料と界面活性剤と保湿剤とバイオサイドとを典型的に含む水性インク。 最新のインク染料は高い耐水性と耐光性とを有する。	環境に優しい。 においが無い。	乾くのが遅い。 腐食する。 紙でにじむ。 黒線が現われるかも知れない。 紙が皺になる。	ほとんどの存在するインクジェット IJ シリーズインクジェットの全て シルバールック, EP0771 658 A2、及び関連特許出願
水性、顔料	水と顔料と界面活性剤と保湿剤とバイオサイドとを典型的に含む水性インク。 顔料は、にじみやウィッキングや黒い線を減らせる点でメリットがある。	環境に優しい。 においが無い。 にじみが少ない。 ウィッキングが少ない。 黒線の発生が少ない。	乾くのが遅い。 腐食する。 顔料がノズルを詰まらせるかも知れない。 顔料がアクチュエータ機構を詰まらせるかも知れない。 紙が皺になる。	IJ02, IJ04, IJ21, IJ26 IJ27, IJ30 シルバールック, EP0771 658 A2、及び関連特許出願 ピエゾエレクトリックインクジェット サーマルインクジェット (重要な制約を伴った)
メチルエチルケトン (MEK)	MEKは、アルミ缶のように難しい表面に工業的にプリントするために使用される高揮発性用材である。	かなり早く乾く。 金属やプラスチックのような様々な素地にプリントできる。	臭気がある。 燃えやすい。	全ての IJ シリーズのインクジェット
アルコール (エタノール、2-ブタノール、他)	アルコールインクは、水の凝固点より低い温度でプリンターを稼動しなければならない場合に使用することができる。この例は、非公開の民生の写真プリントである。	早く乾く。 氷点下の温度で稼動される。 紙の皺は減少。 安価。	少し臭気がある。 燃えやすい。	全ての IJ シリーズのインクジェット

30

【1 9 1 3】

【表 2 4 2】

相変化（ホットメルト）	インクは室温では固体であって、吐出される前にプリントヘッド内で溶融される。ホットメルト・インクは通常はワックスベースで、80℃前後の溶融点を有する。吐出された後、インクは、プリントメディアや転写ローラーと接触してほとんど瞬間的に固まる。	乾燥時間は不要。インクはプリントメディアの表面に瞬時に凍る。ほとんどのプリントメディアでも使用できる。紙の皺の発生は無い。ウィッキングの発生が無い。にじみの発生が無い。黒線の発生が無い。	高い粘性。 プリントされたインクは通常はワックスのような感じである。プリントされたページは‘ブロック’されるかも知れない。インクの温度は永久磁石のキュリー点よりも高いかも知れない。インクヒーターは電力を使う。ウォームアップ時間が長い。	テクトロニクスホットメルト・ピエゾエレクトリック・インクジェット 1989, Nowak, USP 4,820,346 全てのIJシリーズのインクジェット
オイル	オイルインクはオフセット印刷に広く使用される。そのようなオイルインクは、紙の見栄えを良くする（特に、ウィッキングや皺の無い）というメリットを有する。オイルに可溶性の染料や顔料が必要となる。	幾つかの染料にとって、メディアへの溶解性は高い。皺の紙ではない。 （毛管作用により）インクが紙を通り抜けてしまうようなことは無い。	粘性が高い：通常は低粘性が要求されるのであって、高粘性は、インクジェットへの使用にとって重大な制限となる。いくつかの短連鎖、あるいは多分岐オイルは十分に低い粘性である。乾くのが遅い。	全てのIJシリーズのインクジェット
マイクロエマルジョン	マイクロエマルジョンは、安定して、オイルや水や界面活性剤のエマルジョンを自己形成する。 特徴のある滴のサイズは100nmより小さく、界面活性剤の優先曲率 (preferred curvature) により決定される。	インクのにじみ無し。 染料の溶解度は高い。 水、油、及び両親媒性の可溶性染料を使用できる。 顔料の懸濁液を安定にできる。	粘性は水よりも高い。 価格は、水ベースのインクよりも僅かに高い。 高濃度の界面活性剤が必要である（5%前後）。	全てのIJシリーズのインクジェット

10

インクジェットプリンティング

【1914】

相当数の新型のインクジェットプリンタが、好適な実施形態として、インクジェット技術に取って代わるために発展してきている。インクジェットデバイスの様々な組み合わせが、本発明の一部として開示されているプリンタデバイスの中に入めることができる。相互参照表により特別に示される、これらのインクジェットに関連するオーストラリアの仮特許の明細書がある。すなわち、

20

【表 2 4 3】

オーストラリアの 仮番号	出願日	タイトル
P08066	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ01)
P08072	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ02)
P08040	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ03)
P08071	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ04)
P08047	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ05)
P08035	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ06)
P08044	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ07)
P08063	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ08)
P08057	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ90)
P08056	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ10)
P08069	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ11)
P08049	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ12)
P08036	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ13)
P08048	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ14)
P08070	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ15)
P08067	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ16)
P08001	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ17)
P08038	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ18)
P08033	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ19)
P08002	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ20)
P08068	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ21)
P08062	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ22)
P08034	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ23)
P08039	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ24)

10

20

30

【表 2 4 4】

P08041	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ25)
P08004	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ26)
P08037	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ27)
P08043	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ28)
P08042	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ29)
P08064	9 7 年 7 月 1 5 日	画像形成方法及び装置 (IJ30)
P09389	9 7 年 9 月 2 3 日	画像形成方法及び装置 (IJ31)
P09391	9 7 年 9 月 2 3 日	画像形成方法及び装置 (IJ32)
PP0888	9 7 年 1 2 月 1 2 日	画像形成方法及び装置 (IJ33)
PP0891	9 7 年 1 2 月 1 2 日	画像形成方法及び装置 (IJ34)
PP0890	9 7 年 1 2 月 1 2 日	画像形成方法及び装置 (IJ35)
PP0873	9 7 年 1 2 月 1 2 日	画像形成方法及び装置 (IJ36)
PP0993	9 7 年 1 2 月 1 2 日	画像形成方法及び装置 (IJ37)
PP0890	9 7 年 1 2 月 1 2 日	画像形成方法及び装置 (IJ38)
PP1398	9 8 年 1 月 1 9 日	画像形成方法及び装置 (IJ39)
PP2592	9 8 年 3 月 2 5 日	画像形成方法及び装置 (IJ40)
PP2593	9 8 年 3 月 2 5 日	画像形成方法及び装置 (IJ41)
PP3991	9 8 年 6 月 9 日	画像形成方法及び装置 (IJ42)
PP3987	9 8 年 6 月 9 日	画像形成方法及び装置 (IJ43)
PP3985	9 8 年 6 月 9 日	画像形成方法及び装置 (IJ44)
PP3983	9 8 年 6 月 9 日	画像形成方法及び装置 (IJ45)

10

20

30

インクジェットの生産

【1 9 1 5】

さらに、本出願は、多数のインクジェットプリンタの構築に、高度な半導体製造技術が使用されるかも知れない。好適な製造技術は、次の相互参照表に示されているオーストラリア仮特許の明細書中に開示されている。

【表 2 4 5】

オーストラリアの 仮番号	出願日	タイトル
P07935	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM01)
P07936	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM02)
P07937	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM03)
P08061	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM04)

40

【表 2 4 6】

P08054	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM05)
P08065	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM06)
P08055	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM07)
P08053	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM08)
P08078	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM09)
P07933	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM10)
P07950	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM11)
P07949	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM12)
P08060	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM13)
P08059	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM14)
P08073	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM15)
P08076	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM16)
P08075	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM17)
P08079	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM18)
P08050	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM19)
P08052	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM20)
P07948	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM21)
P07951	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM22)
P08074	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM23)
P07941	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM24)
P08077	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM25)
P08058	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM26)
P08051	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM27)
P08045	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM28)
P07952	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM29)
P08046	9 7 年 7 月 1 5 日	画像形成装置の製造方法 (IJM30)
P08503	9 7 年 8 月 1 1 日	画像形成装置の製造方法 (IJM30a)
P09390	9 7 年 9 月 2 3 日	画像形成装置の製造方法 (IJM31)
P09392	9 7 年 9 月 2 3 日	画像形成装置の製造方法 (IJM32)
PP0889	9 7 年 1 2 月 1 2 日	画像形成装置の製造方法 (IJM35)
PP0887	9 7 年 1 2 月 1 2 日	画像形成装置の製造方法 (IJM36)
PP0882	9 7 年 1 2 月 1 2 日	画像形成装置の製造方法 (IJM37)
PP0874	9 7 年 1 2 月 1 2 日	画像形成装置の製造方法 (IJM38)
PP1396	9 8 年 1 月 1 9 日	画像形成装置の製造方法 (IJM39)
PP2591	9 8 年 3 月 2 5 日	画像形成装置の製造方法 (IJM41)

10

20

30

40

【表 2 4 7】

PP3989	9 8 年 6 月 9 日	画像形成装置の製造方法 (IJM40)
PP3990	9 8 年 6 月 9 日	画像形成装置の製造方法 (IJM42)
PP3986	9 8 年 6 月 9 日	画像形成装置の製造方法 (IJM43)
PP3984	9 8 年 6 月 9 日	画像形成装置の製造方法 (IJM44)
PP3982	9 8 年 6 月 9 日	画像形成装置の製造方法 (IJM45)

流体の供給

10

【1 9 1 6】

さらに、本実施例は、インクジェットヘッドへのインク配送システムを利用しても良い。一連のインクジェットノズルにインクを供給するための配送システムは、次の相互参照表に示されているオーストラリア仮特許の明細書中に開示されている。

【表 2 4 8】

オースト ラリアの 仮番号	出願日	タイトル
P08003	9 7 年 7 月 1 5 日	供給方法及び装置 (F 1)
P08005	9 7 年 7 月 1 5 日	供給方法及び装置 (F 2)
P09404	9 7 年 9 月 2 3 日	装置及び方法 (F 3)

20

MEMS 技術

【1 9 1 7】

さらに、本願は、インクジェットプリンタの配列を行う際に高度な半導体マイクロエレクトロメカニカル技術 (MEMS 技術) を利用しても良い。好適なマイクロエレクトロメカニカル技術は、次の相互参照表に示されているオーストラリア仮特許の明細書中に開示されている。

【表 2 4 9】

30

オースト ラリアの 仮番号	出願日	タイトル
P07943	9 7 年 7 月 1 5 日	装置 (MEMS 0 1)
P08006	9 7 年 7 月 1 5 日	装置 (MEMS 0 2)
P08007	9 7 年 7 月 1 5 日	装置 (MEMS 0 3)
P08008	9 7 年 7 月 1 5 日	装置 (MEMS 0 4)
P08010	9 7 年 7 月 1 5 日	装置 (MEMS 0 5)

40

【表 2 5 0】

P08011	97年7月15日	装置 (MEMS06)
P07947	97年7月15日	装置 (MEMS07)
P07945	97年7月15日	装置 (MEMS08)
P07944	97年7月15日	装置 (MEMS09)
P07946	97年7月15日	装置 (MEMS10)
P09393	97年9月23日	装置及び方法 (MEMS11)
PP0875	97年12月12日	装置 (MEMS12)
PP0894	97年12月12日	装置及び方法 (MEMS13)

10

I R 技術

【1918】

さらに、本願は、次の相互参照表に示されているオーストラリア仮特許の明細書中に開示されているような使い捨てカメラシステムを利用するようにしても良い。

【表 2 5 1】

オーストラ リアの仮番 号	出願日	タイトル
PP0895	97年12月12日	画像形成方法及び装置 (IR01)
PP0870	97年12月12日	装置及び方法 (IR02)
PP0869	97年12月12日	装置及び方法 (IR04)
PP0887	97年12月12日	画像形成方法及び装置 (IR05)
PP0885	97年12月12日	画像生成システム (IR06)
PP0884	97年12月12日	画像形成方法及び装置 (IR10)
PP0886	97年12月12日	画像形成方法及び装置 (IR12)
PP0871	97年12月12日	装置及び方法 (IR13)
PP0876	97年12月12日	画像処理方法及び装置 (IR14)
PP0877	97年12月12日	装置及び方法 (IR16)
PP0878	97年12月12日	装置及び方法 (IR17)
PP0879	97年12月12日	装置及び方法 (IR18)
PP0883	97年12月12日	装置及び方法 (IR19)
PP0880	97年12月12日	装置及び方法 (IR20)
PP0881	97年12月12日	装置及び方法 (IR21)

20

30

40

ドットカード技術

【1919】

さらに、本願は、次の相互参照表に示されているオーストラリア仮特許の明細書中に開示されているようなデータ配信システムを使用するようにしても良い。

【表 2 5 2】

オーストラリアの 仮番号	出願日	タイトル
PP2370	98年3月16日	データ処理方法及び装置 (D o t 0 1)
PP2371	98年3月16日	データ処理方法及び装置 (D o t 0 2)

アートカム技術

10

【1 9 2 0】

さらに、本願は、次の相互参照表に示されているオーストラリア仮特許の明細書中に開示されているアートカム型装置のようなカメラとデータ処理技術を使用するようにしても良い。

【表 2 5 3】

オーストラリアの 仮番号	出願日	タイトル
P07991	97年7月15日	画像処理方法及び装置 (A R T 0 1)
P08505	97年8月11日	画像処理方法及び装置 (A R T 0 1 a)
P07988	97年7月15日	画像処理方法及び装置 (A R T 0 2)
P07993	97年7月15日	画像処理方法及び装置 (A R T 0 3)
P08012	97年7月15日	画像処理方法及び装置 (A R T 0 5)
P08017	97年7月15日	画像処理方法及び装置 (A R T 0 6)
P08014	97年7月15日	メディアデバイス (A R T 0 7)
P08025	97年7月15日	画像処理方法及び装置 (A R T 0 8)
P08032	97年7月15日	画像処理方法及び装置 (A R T 0 9)
P07999	97年7月15日	画像処理方法及び装置 (A R T 1 0)
P07998	97年7月15日	画像処理方法及び装置 (A R T 1 1)
P08031	97年7月15日	画像処理方法及び装置 (A R T 1 2)
P08030	97年7月15日	メディアデバイス (A R T 1 3)
P08498	97年8月11日	画像処理方法及び装置 (A R T 1 4)
P07997	97年7月15日	メディアデバイス (A R T 1 5)
P07979	97年7月15日	メディアデバイス (A R T 1 6)
P08015	97年7月15日	メディアデバイス (A R T 1 7)

20

30

40

【表 2 5 4】

P07978	97年7月15日	メディアデバイス (ART 18)
P07982	97年7月15日	データ処理方法及び装置 (ART 19)
P07989	97年7月15日	データ処理方法及び装置 (ART 20)
P08019	97年7月15日	メディア処理方法及び装置 (ART 21)
P07980	97年7月15日	画像処理方法及び装置 (ART 22)
P07942	97年7月15日	画像処理方法及び装置 (ART 23)
P08018	97年7月15日	画像処理方法及び装置 (ART 24)
P07938	97年7月15日	画像処理方法及び装置 (ART 25)
P08016	97年7月15日	画像処理方法及び装置 (ART 26)
P08024	97年7月15日	画像処理方法及び装置 (ART 27)
P07940	97年7月15日	データ処理方法及び装置 (ART 28)
P07939	97年7月15日	データ処理方法及び装置 (ART 29)
P08501	97年8月11日	画像処理方法及び装置 (ART 30)
P08500	97年8月11日	画像処理方法及び装置 (ART 31)
P07987	97年7月15日	データ処理方法及び装置 (ART 32)
P08022	97年7月15日	画像処理方法及び装置 (ART 33)
P08497	97年8月11日	画像処理方法及び装置 (ART 30)
P08029	97年7月15日	センサー形成方法及び装置 (ART 36)
P07985	97年7月15日	データ処理方法及び装置 (ART 37)
P08020	97年7月15日	データ処理方法及び装置 (ART 38)
P08023	97年7月15日	データ処理方法及び装置 (ART 39)
P09395	97年9月23日	データ処理方法及び装置 (ART 4)
P08021	97年7月15日	データ処理方法及び装置 (ART 40)
P08504	97年8月11日	画像処理方法及び装置 (ART 42)
P08000	97年7月15日	データ処理方法及び装置 (ART 43)
P07977	97年7月15日	データ処理方法及び装置 (ART 44)
P07934	97年7月15日	データ処理方法及び装置 (ART 45)
P07990	97年7月15日	データ処理方法及び装置 (ART 46)
P08499	97年8月11日	画像処理方法及び装置 (ART 47)
P08502	97年8月11日	画像処理方法及び装置 (ART 48)
P07981	97年7月15日	データ処理方法及び装置 (ART 50)
P07986	97年7月15日	データ処理方法及び装置 (ART 51)
P07983	97年7月15日	データ処理方法及び装置 (ART 52)
P08026	97年7月15日	画像処理方法及び装置 (ART 53)
P08027	97年7月15日	画像処理方法及び装置 (ART 54)

10

20

30

40

【表 2 5 5】

P08028	9 7 年 7 月 1 5 日	画像処理方法及び装置 (ART 5 6)
P09394	9 7 年 9 月 2 3 日	画像処理方法及び装置 (ART 5 7)
P09396	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 5 8)
P09397	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 5 9)
P09398	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 6 0)
P09399	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 6 1)
P09400	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 6 2)
P09401	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 6 3)
P09402	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 6 4)
P09403	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 6 5)
P09405	9 7 年 9 月 2 3 日	データ処理方法及び装置 (ART 6 6)
PP0959	9 7 年 1 2 月 1 6 日	データ処理方法及び装置 (ART 6 8)
PP1397	9 8 年 1 月 1 9 日	メディアデバイス (ART 6 9)

10

【1 9 2 1】

20

当業者にとって、特定の実施例に示された本発明に対して、広範に述べられた発明の範囲及び精神から離脱することなく多様な変形や変更を加えることが可能である。本実施例は、従って例外的なものであり、制限的なものではないものとあらゆる面で考慮すべきである。

【図面の簡単な説明】

【1 9 2 2】

【図 1】 好適な実施例によって構成されたアートカム装置の説明図である。

【図 2】 主要なアートカム電子コンポーネントの概略ブロック図である。

【図 3】 アートカム中央プロセッサの概略ブロック図である。

【図 3 (a)】 V L I Wベクトルプロセッサの詳細図である。

30

【図 4】 処理ユニットの詳細図である。

【図 5】 A L U 1 8 8 の詳細図である。

【図 6】 入力ブロックの詳細図である。

【図 7】 出力ブロックの詳細図である。

【図 8】 レジスタブロックの詳細図である。

【図 9】 クロスバー 1 の詳細図である。

【図 1 0】 クロスバー 2 の詳細図である。

【図 1 1】 読み出しプロセスブロックの詳細図である。

【図 1 2】 読み出しプロセスブロックの詳細図である。

【図 1 3】 バレルシフタブロックの詳細図である。

40

【図 1 4】 アダー／ロジックブロックの詳細図である。

【図 1 5】 乗算ブロックの詳細図である。

【図 1 6】 I / O アドレス生成器の詳細図である。

【図 1 7】 画素蓄積フォーマットの説明図である。

【図 1 8】 順次読み出しイタレータプロセスの説明図である。

【図 1 9】 ボックス読み出しイタレータプロセスの説明図である。

【図 2 0】 ボックス書き込みイタレータプロセスの説明図である。

【図 2 1】 垂直ストリップ読み出し／書き込みイタレータプロセスの説明図である。

【図 2 2】 垂直ストリップ読み出し／書き込みイタレータプロセスの説明図である。

【図 2 3】 シーケンシャル生成プロセスの説明図である。

50

- 【図 2 4】シーケンシャル生成プロセスの説明図である。
- 【図 2 5】垂直ストリップ生成プロセスの説明図である。
- 【図 2 6】垂直ストリップ生成プロセスの説明図である。
- 【図 2 7】画素データ構造の説明図である。
- 【図 2 8】画素処理プロセスの説明図である。
- 【図 2 9】表示コントローラの概略ブロック図である。
- 【図 3 0】C C D 像編成の説明図である。
- 【図 3 1】論理的画像用の蓄積フォーマットの説明図である。
- 【図 3 2】内部画像メモリ蓄積フォーマットの説明図である。
- 【図 3 3】画像ピラミッド蓄積フォーマットの説明図である。 10
- 【図 3 4】アートカードをサンプリングするプロセスの時系列の説明図である。
- 【図 3 5】スーパーサンプリングプロセスの説明図である。
- 【図 3 6】回転したアートカードを読み取るプロセスの説明図である。
- 【図 3 7】アートカードを復号化するために必要なステップのフローチャートである。
- 【図 3 8】単一のアートカードの左隅の拡大図である。
- 【図 3 9】検出用の単一ターゲットである。
- 【図 4 0】ターゲットを検出するため使用される方法の説明図である。
- 【図 4 1】二つのターゲット間の距離を計算する方法の説明図である。
- 【図 4 2】重心ドリフトのプロセスの説明図である。
- 【図 4 3】重心ルックアップテーブルの一形態を示す図である。 20
- 【図 4 4】重心更新プロセスの説明図である。
- 【図 4 5】好適な実施例において利用されるデルタ処理用ルックアップテーブルの説明図である。
- 【図 4 6】アートカードのスクランブルを解除するプロセスの説明図である。
- 【図 4 7】ドットの系列の拡大図である。
- 【図 4 8】ドットカードのデータ面の説明図である。
- 【図 4 9】単一データブロックの概略説明図である。
- 【図 5 0】単一データブロックの説明図である。
- 【図 5 1】図 5 0 のデータブロックの部分拡大図である。
- 【図 5 2】図 5 0 のデータブロックの部分拡大図である。 30
- 【図 5 3】単一ターゲット構造の説明図である。
- 【図 5 4】データブロックのターゲット構造の説明図である。
- 【図 5 5】データ領域のボーダークロック用領域に対するターゲットの位置関係の説明図である。
- 【図 5 6】データブロックのオリエンテーションカラムの説明図である。
- 【図 5 7】データブロックのドットのアレイの説明図である。
- 【図 5 8】リード・ソロモン符号化のデータの構造の概略説明図である。
- 【図 5 9】例示的なリード・ソロモン符号化プロセスの説明図である。
- 【図 6 0】リード・ソロモン符号化プロセスの説明図である。
- 【図 6 1】データブロック内の符号化されたデータのレイアウトの説明図である。 40
- 【図 6 2】代替的なアートカードをサンプリングする際のサンプリングプロセスの説明図である。
- 【図 6 3】回転した代替的なアートカードをサンプリングする一例の拡大図である。
- 【図 6 4】スキャニングプロセスの説明図である。
- 【図 6 5】スキャニングプロセスの起こり得るスキャニング分布の説明図である。
- 【図 6 6】シンボル誤りとリード・ソロモンブロック誤りの確率の間の関係を示す図である。
- 【図 6 7】復号化プロセスのフローチャートである。
- 【図 6 8】復号化プロセスのプロセス稼働図である。
- 【図 6 9】復号化におけるデータフローステップの説明図である。 50

- 【図 7 0】読み出しプロセスの詳細図である。
- 【図 7 1】代替的なアートカードの開始を検出するプロセスの詳細図である。
- 【図 7 2】ビットデータ抽出プロセスの詳細図である。
- 【図 7 3】復号化プロセスで利用されるセグメンテーションプロセスの説明図である。
- 【図 7 4】ターゲットを見つける復号化プロセスの詳細図である。
- 【図 7 5】ターゲットを見つける際に利用されるデータ構造の説明図である。
- 【図 7 6】L a n c o s 3 関数構造の説明図である。
- 【図 7 7】クロックマーク及びボーダー領域を示すデータブロックの部分拡大図である。
- 【図 7 8】ビットイメージを復号化する際の処理ステップの説明図である。
- 【図 7 9】ビットイメージを復号化する際のデータフローステップの説明図である。 10
- 【図 8 0】好適な実施例のスクランブル解除プロセスの説明図である。
- 【図 8 1】コンボルバの一実施形態の説明図である。
- 【図 8 2】コンボルブプロセスの説明図である。
- 【図 8 3】コンポジットプロセスの説明図である。
- 【図 8 4】通常のコンポジットプロセスの詳細図である。
- 【図 8 5】ワーブマップを使用するワーブプロセスの説明図である。
- 【図 8 6】ワーブバイリニア補間プロセスの説明図である。
- 【図 8 7】スパン計算のプロセスの説明図である。
- 【図 8 8】基本スパン計算プロセスの説明図である。
- 【図 8 9】スパン計算プロセスの詳細な一実施形態の説明図である。 20
- 【図 9 0】画像ピラミッドレベルを読み取るプロセスの説明図である。
- 【図 9 1】バイリニア補間のためピラミッドテーブルを使用する方法の説明図である。
- 【図 9 2】ヒストグラム収集プロセスの説明図である。
- 【図 9 3】色変換プロセスの説明図である。
- 【図 9 4】色変換プロセスの説明図である。
- 【図 9 5】色空間変換プロセスの詳細図である。
- 【図 9 6】入力座標を計算するプロセスの説明図である。
- 【図 9 7】フィードバック付きコンポジット処理するプロセスの説明図である。
- 【図 9 8】汎用スケーリングプロセスの説明図である。
- 【図 9 9】Xスケーリングプロセスにおけるスケールの説明図である。 30
- 【図 1 0 0】Yスケーリングプロセスにおけるスケールの説明図である。
- 【図 1 0 1】モザイクプロセスの説明図である。
- 【図 1 0 2】サブピクセル移動プロセスの説明図である。
- 【図 1 0 3】コンポジットプロセスの説明図である。
- 【図 1 0 4】フィードバック付きでコンポジットするプロセスの説明図である。
- 【図 1 0 5】入力画像からの色でタイル貼りするプロセスの説明図である。
- 【図 1 0 6】フィードバック付きでタイル貼りするプロセスの説明図である。
- 【図 1 0 7】テクスチャ置換でタイル貼りするプロセスの説明図である。
- 【図 1 0 8】入力画像からの色でタイル貼りするプロセスの説明図である。
- 【図 1 0 8】入力画像からの色でタイル貼りするプロセスの説明図である。 40
- 【図 1 0 9】フィードバック無しでテクスチャを与えるプロセスの説明図である。
- 【図 1 1 0】フィードバック付きでテクスチャを与えるプロセスの説明図である。
- 【図 1 1 1】C C D 画素の回転のプロセスの説明図である。
- 【図 1 1 2】緑色サブピクセルの補間のプロセスの説明図である。
- 【図 1 1 3】青色サブピクセルの補間のプロセスの説明図である。
- 【図 1 1 4】赤色サブピクセルの補間のプロセスの説明図である。
- 【図 1 1 5】奇数画素行に対し 0 度回転の C C D 画素補間のプロセスの説明図である。
- 【図 1 1 6】偶数画素行に対し 0 度回転の C C D 画素補間のプロセスの説明図である。
- 【図 1 1 7】L a b 色空間への色変換のプロセスの説明図である。
- 【図 1 1 8】 $1/\sqrt{X}$ の計算のプロセスの説明図である。 50

- 【図 1 1 9】 $1/\sqrt{X}$ の計算の実施例の詳細説明図である。
- 【図 1 2 0】 バンプマップによる法線計算のプロセスの説明図である。
- 【図 1 2 1】 バンプマップによる照度計算のプロセスの説明図である。
- 【図 1 2 2】 バンプマップによる照度計算のプロセスの詳細説明図である。
- 【図 1 2 3】 指向性ライトを使用する L の計算のプロセスの説明図である。
- 【図 1 2 4】 オムニライト及びスポットライトを使用する L の計算のプロセスの説明図である。
- 【図 1 2 5】 オムニライト及びスポットライトを使用する L の計算の一実施形態の説明図である。
- 【図 1 2 6】 N・L 内積を計算するプロセスの説明図である。 10
- 【図 1 2 7】 N・L 内積を計算するプロセスの詳細説明図である。
- 【図 1 2 8】 R・V 内積を計算するプロセスの説明図である。
- 【図 1 2 9】 R・V 内積を計算するプロセスの詳細説明図である。
- 【図 1 3 0】 減衰計算の入力及び出力の説明図である。
- 【図 1 3 1】 減衰計算の実際の実施形態の説明図である。
- 【図 1 3 2】 円錐ファクタのグラフの説明図である。
- 【図 1 3 3】 半影計算のプロセスの説明図である。
- 【図 1 3 4】 半影計算に利用される角の説明図である。
- 【図 1 3 5】 半影計算への入力及び出力の説明図である。
- 【図 1 3 6】 半影計算の実際の実施形態の説明図である。 20
- 【図 1 3 7】 周辺光計算への入力及び出力の説明図である。
- 【図 1 3 8】 周辺光計算の実際の実施形態の説明図である。
- 【図 1 3 9】 散光計算の実際の実施形態の説明図である。
- 【図 1 4 0】 散光計算への入力及び出力の説明図である。
- 【図 1 4 1】 散光計算の実際の実施形態の説明図である。
- 【図 1 4 2】 反射計算への入力及び出力の説明図である。
- 【図 1 4 3】 反射計算の実際の実施形態の説明図である。
- 【図 1 4 4】 反射計算への入力及び出力の説明図である。
- 【図 1 4 5】 反射計算の実際の実施形態の説明図である。
- 【図 1 4 6】 周辺光だけの独計算の実際の実施形態の説明図である。 30
- 【図 1 4 7】 光計算のプロセス概要の説明図である。
- 【図 1 4 8】 単一无限光源用の照明計算の一例の説明図である。
- 【図 1 4 9】 バンプマップを用いないオムニ光源用の照明計算の一例の説明図である。
- 【図 1 5 0】 バンプマップを用いるオムニ光源用の照明計算の一例の説明図である。
- 【図 1 5 1】 バンプマップを用いないスポットライト光源用の照明計算の一例の説明図である。
- 【図 1 5 2】 関連したバンプマップを用いて画像に単一スポットライトを当てるプロセスの説明図である。
- 【図 1 5 3】 単一プリントヘッドの論理的レイアウトの説明図である。
- 【図 1 5 4】 プリントヘッドインタフェースの構造の説明図である。 40
- 【図 1 5 5】 Lab 画像の回転のプロセスの説明図である。
- 【図 1 5 6】 印刷画像の画素のフォーマットの説明図である。
- 【図 1 5 7】 ディザリングプロセスの説明図である。
- 【図 1 5 8】 8ビットドット出力を生成するプロセスの説明図である。
- 【図 1 5 9】 カードリーダーの斜視図である。
- 【図 1 6 0】 カードリーダーの展開斜視図である。
- 【図 1 6 1】 アートカードリーダーのクローズアップ図である。
- 【図 1 6 2】 プリントロール及びプリントヘッドの斜視図である。
- 【図 1 6 3】 プリントロールの第 1 の展開斜視図である。
- 【図 1 6 4】 プリントロールの第 2 の展開斜視図である。 50

- 【図 1 6 5】プリントロール認証チップの説明図である。
- 【図 1 6 6】プリントロール認証チップの拡大図である。
- 【図 1 6 7】一段階認証チップデータプロトコルの説明図である。
- 【図 1 6 8】二段階認証チップデータプロトコルの説明図である。
- 【図 1 6 9】第 1 のプレゼンスオンリープロトコルの説明図である。
- 【図 1 7 0】第 2 のプレゼンスオンリープロトコルの説明図である。
- 【図 1 7 1】第 3 のデータプロトコルの説明図である。
- 【図 1 7 2】第 4 のデータプロトコルの説明図である。
- 【図 1 7 3】最長期間 L F S R の概略ブロック図である。
- 【図 1 7 4】クロック制限フィルターの概略ブロック図である。 10
- 【図 1 7 5】タンパー検出ラインの概略ブロック図である。
- 【図 1 7 6】特大の n M O S トランジスタの概略ブロック図である。
- 【図 1 7 7】タンパー検出ラインからの多数の X O R の取り出しを説明する図である。
- 【図 1 7 8】タンパーラインがノイズ発生回路を覆う状態の説明図である。
- 【図 1 7 9】通常の F E T の実施形態の説明図である。
- 【図 1 8 0】好適な実施例の修正された F E T の実施形態の説明図である。
- 【図 1 8 1】認証チップの略ブロックである。
- 【図 1 8 2】メモリマップの一例の説明図である。
- 【図 1 8 3】定数メモリマップの一例の説明図である。
- 【図 1 8 4】R A M メモリマップの一例の説明図である。 20
- 【図 1 8 5】フラッシュメモリ変数メモリマップの一例の説明図である。
- 【図 1 8 6】フラッシュメモリプログラムメモリマップの一例の説明図である。
- 【図 1 8 7】状態機械のコンポーネント間のデータフロー及び関係を示す図である。
- 【図 1 8 8】I / O ユニットのコンポーネント間のデータフロー及び関係を示す図である。
- 。
 - 【図 1 8 9】算術論理ユニット (A L U) の概略ブロック図である。
 - 【図 1 9 0】R P L ユニットの概略ブロック図である。
 - 【図 1 9 1】A L U の R O R ブロックの概略ブロック図である。
 - 【図 1 9 2】プログラムカウンタユニットのブロック図である。
 - 【図 1 9 3】メモリーユニットの概略ブロック図である。 30
 - 【図 1 9 4】アドレス生成ユニットの概略ブロック図である。
 - 【図 1 9 5】J S I G E N ユニットの概略ブロック図である。
 - 【図 1 9 6】J S R G E N ユニットの概略ブロック図である。
 - 【図 1 9 7】D B R G E N ユニットの概略ブロック図である。
 - 【図 1 9 8】L D K G E N ユニットの概略ブロック図である。
 - 【図 1 9 9】R P L G E N ユニットの概略ブロック図である。
 - 【図 2 0 0】V A R G E N ユニットの概略ブロック図である。
 - 【図 2 0 1】C L R G E N ユニットの概略ブロック図である。
 - 【図 2 0 2】B I T G E N ユニットの概略ブロック図である。
 - 【図 2 0 3】プリントロール認証チップに記憶された情報を示す図である。 40
 - 【図 2 0 4】アートカム認証チップ内に記憶されたデータの説明図である。
 - 【図 2 0 5】プリントヘッドパルス特徴付けプロセスの説明図である。
 - 【図 2 0 6】プリントヘッドインク供給機構の拡大断面斜視図である。
 - 【図 2 0 7】インクヘッド供給ユニットの底面斜視図である。
 - 【図 2 0 8】インクヘッド供給ユニットの底面側の断面図である。
 - 【図 2 0 9】インクヘッド供給ユニットの上面斜視図である。
 - 【図 2 1 0】インクヘッド供給ユニットの上面側の断面図である。
 - 【図 2 1 1】プリントヘッドの一部分の斜視図である。
 - 【図 2 1 2】プリントヘッドユニットの拡大斜視図である。
 - 【図 2 1 3】アートカムカメラの内部の上面側の斜視図であり、平らにされた部品が示さ 50

れている。

【図 2 1 4】アートカムカメラの内部の底面側の斜視図であり、平らにされた部品が示されている。

【図 2 1 5】アートカムカメラの内部の第 1 の上面側の斜視図であり、アートカムに収容された部品が示されている。

【図 2 1 6】アートカムカメラの内部の第 2 の上面側の斜視図であり、アートカムに収容された部品が示されている。

【図 2 1 7】アートカムカメラの内部の第 2 の上面側の斜視図であり、アートカムに収容された部品が示されている。

【図 2 1 8】葉書プリントロールの裏書き部分の説明図である。

10

【図 2 1 9】画像印刷後の葉書プリントロールの対応した表画像の説明図である。

【図 2 2 0】顧客が購入できる状態のプリントロールの形態の説明図である。

【図 2 2 1】アートカムアプリケーション全体のソフトウェア／ハードウェアモジュールのレイアウトの説明図である。

【図 2 2 2】カメラマネージャのソフトウェア／ハードウェアモジュールのレイアウトの説明図である。

【図 2 2 3】画像処理マネージャのソフトウェア／ハードウェアモジュールのレイアウトの説明図である。

【図 2 2 4】プリンタマネージャのソフトウェア／ハードウェアモジュールのレイアウトの説明図である。

20

【図 2 2 5】画像処理マネージャのソフトウェア／ハードウェアモジュールのレイアウトの説明図である。

【図 2 2 6】ファイルマネージャのソフトウェア／ハードウェアモジュールのレイアウトの説明図である。

【図 2 2 7】代替的な形態のプリントロールの部分断面斜視図である。

【図 2 2 8】図 2 2 7 のプリントロールの左側の拡大斜視図である。

【図 2 2 9】単一プリントロールの右側の拡大斜視図である。

【図 2 3 0】プリントロールのコア部分の部分断面拡大斜視図である。

【図 2 3 1】プリントロールのコア部分の第 2 の拡大斜視図である。

【図 2 3 2】‘ B i z c a r d ’ の正面図である。

30

【図 2 3 3】さらなる改良に従って構成されたカメラシステムを概略的に示す。

【図 2 3 4】出力メディアの表面と裏面に印刷をするためのプリンター装置を概略的に示す。

【図 2 3 5】写真の裏面における出力データの形式を示す。

【図 2 3 6】出力メディアの拡大された部分を示す。

【図 2 3 7】写真の裏面からデータを読み込むために使用される読み取り装置を示す。

【図 2 3 8】さらなる改良の装置の利用を示す。

【図 2 3 9】画像の向きの具体的な処理の概略的な例を示す。

【図 2 4 0】さらなる改良に従って構成されたカメラを使用した場合の画像の特有の効果を作る方法を示す。

40

【図 2 4 1】さらなる改良に従ったプリントロールを示す。

【図 2 4 2】さらに改良された実施例の方法を示す。

【図 2 4 3】さらなる改良の操作方法を示す。

【図 2 4 4】さらなる改良に従った画像処理の一形態を示す。

【図 2 4 5】さらなる改良の操作方法を示す。

【図 2 4 6】画像のキャプチャー及び出力のプロセスを示す。

【図 2 4 7】赤目除去のプロセスを示す。

【図 2 4 8】標準的なアートカム装置に従って構成された写真印刷編成を示す。

【図 2 4 9】さらなる改良に従って構成されたデュアルプリントヘッド編成を示す。

【図 2 5 0】さらなる改良のカール除去装置を示す。

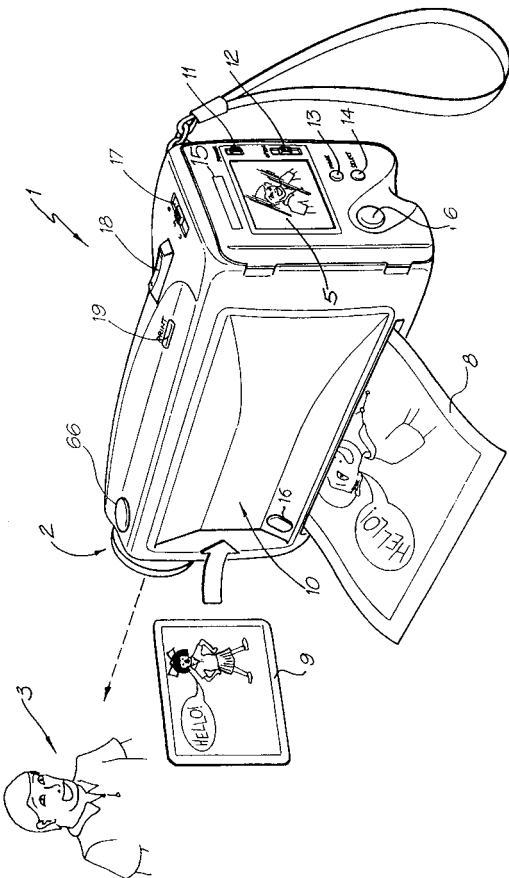
50

- 【図 2 5 1】立体写真画像を見るプロセスを示す。
- 【図 2 5 2】立体写真効果を作成するために設計された、さらなる改善のレンズシステムを示す。
- 【図 2 5 3】立体写真画像の創造を示す部分斜視図である。
- 【図 2 5 4】さらなる改良に従って立体写真画像を作るための装置を示す。
- 【図 2 5 5】図 2 5 4 の位置決めユニットをさらに詳細に示す。
- 【図 2 5 6】立体写真画像の作成に好適なカメラ装置を示す。
- 【図 2 5 7】立体写真効果を改良すべく、不透明な裏当てを使用するプロセスを示す。
- 【図 2 5 8】プリントメディアに画像を生成する方法を概略的に示す。
- 【図 2 5 9】本発明に従って構成されたプリントメディアの構造を示す。 10
- 【図 2 6 0】さらなる改良に従って構成されたプリントメディアの利用を示す。
- 【図 2 6 1】さらなる改良に従ったプリントメディアの構造の第 1 形態を示す。
- 【図 2 6 2】本発明に従ったプリントメディアの構造のさらなる形態を示す。
- 【図 2 6 3】本発明に従ったプリントメディアの構造のさらなる形態の概略的な断面図を示す。
- 【図 2 6 4】本発明に従ったプリントメディアの構造のさらなる形態の概略的な断面図を示す。
- 【図 2 6 5】図 2 6 3 及び 7 に従ったプリントメディア構成の製造の一形態を示す。
- 【図 2 6 6】図 2 6 5 の編成を利用するために繊維材料を押し出すことによる製造の別の形態を示す。 20
- 【図 2 6 7】さらなる改良の主要ステップを示す。
- 【図 2 6 8】さらなる改良にて使用される Sobel フィルターの効果を示す。
- 【図 2 6 9】さらなる改良にて使用される曲線のオフセット方法を示す。
- 【図 2 7 0】さらなる改良にて使用される曲線のオフセット方法を示す。
- 【図 2 7 1】認識された顔を含む標準的な画像を示す。
- 【図 2 7 2】興味の領域を決定するための第 1 のフローチャートを示す。
- 【図 2 7 3】興味の領域を決定するための第 2 のフローチャートを示す。
- 【図 2 7 4】画像をタイリングする最初の工程を示す。
- 【図 2 7 5】画像のタイリングのための別のタイルパターンを示す。
- 【図 2 7 6】タイルの不透明なマスクを示す。 30
- 【図 2 7 7】図 2 7 6 のタイルのための、対応する表面テクスチャ高さを示す。
- 【図 2 7 8】図 2 7 6 のタイルのための、対応する表面テクスチャ高さを示す。
- 【図 2 7 9】画像を修正するために global opacity を使用する工程を示す。
- 【図 2 8 0】実際のペイントを混合させる効果を真似るための、ストローキング効果を修正する方法を示す。
- 【図 2 8 1】ブラッシュストロークのテーブルを作成する工程を示す。
- 【図 2 8 2】ストロークカラーパレットを作成する工程を示す。
- 【図 2 8 3】結果として起こるペインティング工程を示す。
- 【図 2 8 4】ブラッシュストロークを合成するさらなる実施例におけるステップのフローチャートを示す。 40
- 【図 2 8 5】ベジエ曲線の区分的線分への変換方法を示す。
- 【図 2 8 6】ブラッシュストロークプロセスを示す。
- 【図 2 8 7】さらなる実施例で使用される、好適なブラッシュスタンプを示す。
- 【図 2 8 8】さらなる改良の第 1 の編成を示す。
- 【図 2 8 9】さらなる改良の操作のフローチャートを示す。
- 【図 2 9 0】高解像度で印刷された画像の構成を示す。
- 【図 2 9 1】コピーを作成するための、図 2 9 0 の画像を処理するための装置を示す。
- 【図 2 9 2】centroid の処理ステップを示す。
- 【図 2 9 3】一連の検知したパターンを示す。
- 【図 2 9 4】さらなる改良の実施の図を示す。 50

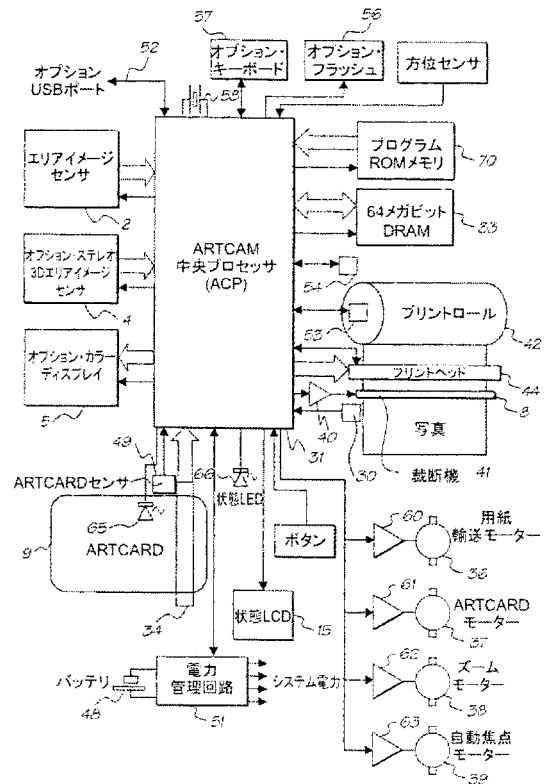
- 【図 2 9 5】さらなる改良を示す。
- 【図 2 9 6】さらなる実施例のカード読み取り装置を示す。
- 【図 2 9 7】ブラシ・バンプマップを示す。
- 【図 2 9 8】ブラシ・バンプマップを示す。
- 【図 2 9 9】背景キャンバスのバンプマップを示す。
- 【図 3 0 0】背景キャンバスのバンプマップを示す。
- 【図 3 0 1】バンプマップを結合するプロセスを示す。
- 【図 3 0 2】バンプマップを結合するプロセスを示す。
- 【図 3 0 3】マップの背景を示す。
- 【図 3 0 4】さらなる改良に従ってバンプマップを結合するプロセスを示す。 10
- 【図 3 0 5】さらなる改良に従ってバンプマップを結合するプロセスを示す。
- 【図 3 0 6】芸術的な画像の生成のさらなる改良の方法におけるステップを示す。
- 【図 3 0 7】一つのガモットを第 2 のガモットにマッピングするプロセスを示す。
- 【図 3 0 8】さらなる改良の一実施形態を示す。
- 【図 3 0 9】ガモットリマッピングの好ましい形態を示す。
- 【図 3 1 0】さらなる改善にて使用されるガモット・モーフィングを示す。
- 【図 3 1 1】アートカム装置の基本操作を示す。
- 【図 3 1 2】さらなる改良で使用される一連のアートカードを示す。
- 【図 3 1 3】さらなる改良で使用されるアルゴリズムのフローチャート図である。
- 【図 3 1 4】本発明に従って作成される、印刷された布の出力の概略図である。 20
- 【図 3 1 5】さらなる改良の接続方式を示す。
- 【図 3 1 6】部分的に断面が描かれた、さらなる改良の斜視図を示す。
- 【図 3 1 7】書き込みデータエリア配列を有するカードを示す。
- 【図 3 1 8】限定された数の書き込みデータ領域のみを有するカードを示す。
- 【図 3 1 9】データ領域の構成を示す。
- 【図 3 2 0】ターゲット構成を示す。
- 【図 3 2 1】さらなる改良の装置を示す。
- 【図 3 2 2】図 3 2 1 を近づいて見た状態を示す。
- 【図 3 2 3】読み取り装置の中にカードを挿入するプロセスを示す。
- 【図 3 2 4】カードを取り出すプロセスを示す。 30
- 【図 3 2 5】カードにデータ領域を書くプロセスを示す。
- 【図 3 2 6】カードリーダーの構造の概略図である。
- 【図 3 2 7】さらなる改善の概略的な配列を示す。
- 【図 3 2 8】さらなる改善のインターフェースの例を示す。
- 【図 3 2 9】さらなる改善のソフトウェアモジュールの構成の一形態を示す。
- 【図 3 3 0】アートカムシステムの操作の概略図である。
- 【図 3 3 1】アートカムシステムの変形された操作の第 1 の例が示されている。
- 【図 3 3 2】アートカムの操作を変更するリピティションカードが示されている。
- 【図 3 3 3】アートカムの操作を変更するためのアートカードテストカードが示されてい 40
- る。
- 【図 3 3 4】アートカム装置の出力テスト結果を示す。
- 【図 3 3 5】さらなる改良に従って構成されたカメラシステムを概略的に示す。
- 【図 3 3 6】さらなる改良のプリンター装置及び録音装置を概略的に示す。
- 【図 3 3 7】写真の裏面の磁気帯の構成を示す。
- 【図 3 3 8】写真の裏面に録音されたデータを読むために使用される読み取り装置を示す。
- 。 【図 3 3 9】さらなる改良の装置の使用を示す。
- 【図 3 4 0】アートカム装置の機能部分の概略図を示す。
- 【図 3 4 1】さらなる改良を使用するステップを示す。
- 【図 3 4 2】さらなる改良に従ったアートカム装置の操作を示す。 50

- 【図 3 4 3】出力された“写真”の裏面に印刷された他の実施例を示す。
 【図 3 4 4】さらなる実施例に従って構成されたプリンター装置の内部を示す。
 【図 3 4 5】さらなる実施例に従って構成されたネットワーク配布システムを示す。
 【図 3 4 6】図 3 4 5 のプリンターコンピュータの操作を概略的に示す。

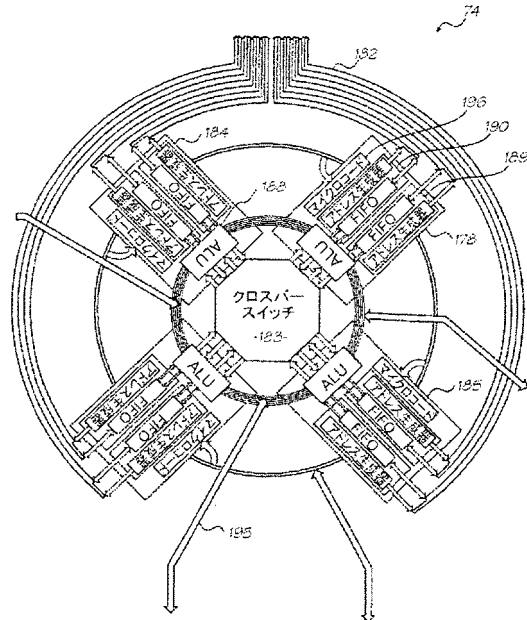
【図 1】



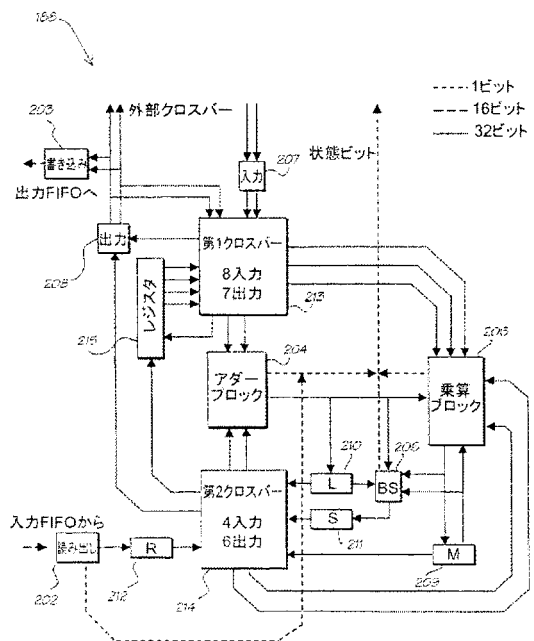
【図 2】



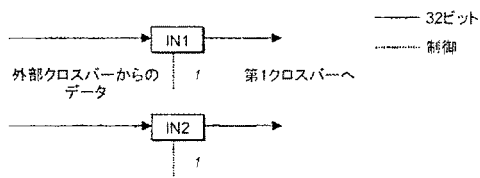
【図 3 (a)】



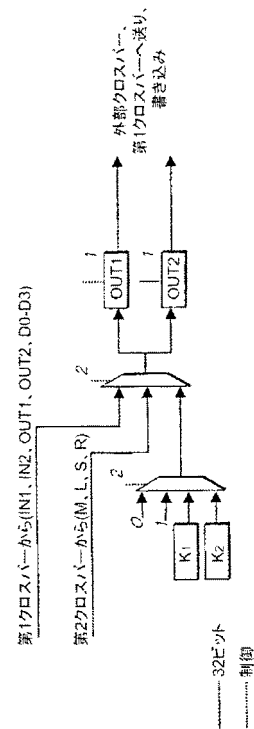
【図 5】



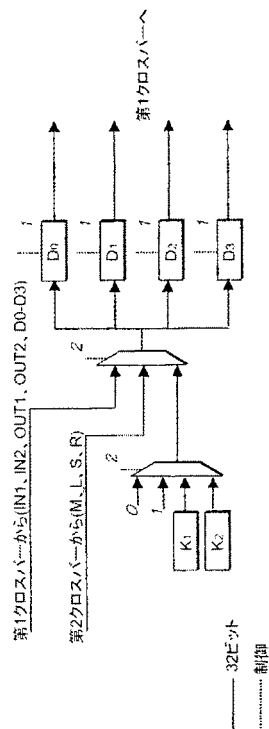
【図 6】



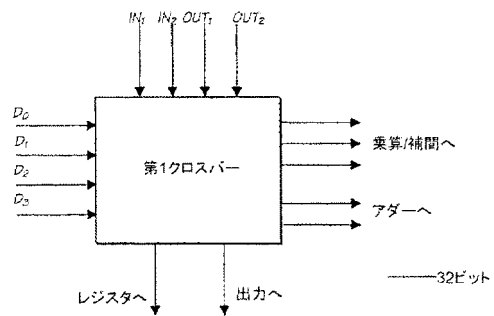
【図 7】



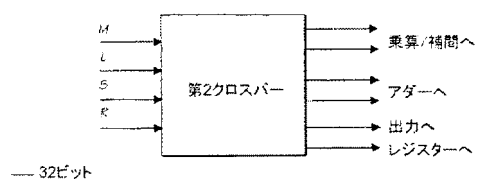
【図 8】



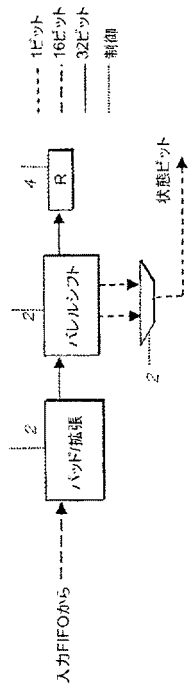
【図 9】



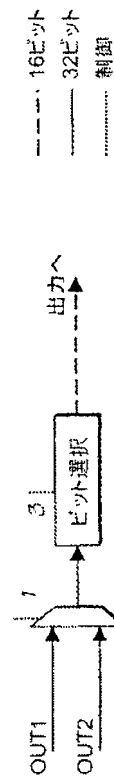
【図 10】



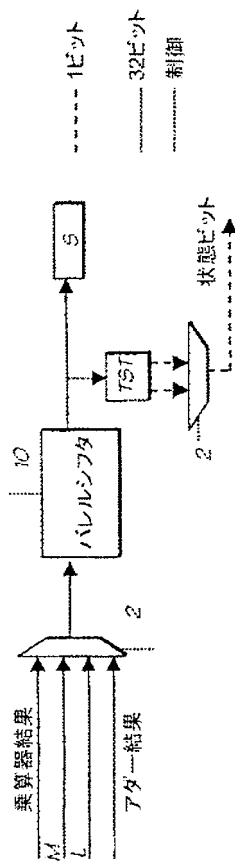
【図 1 1】



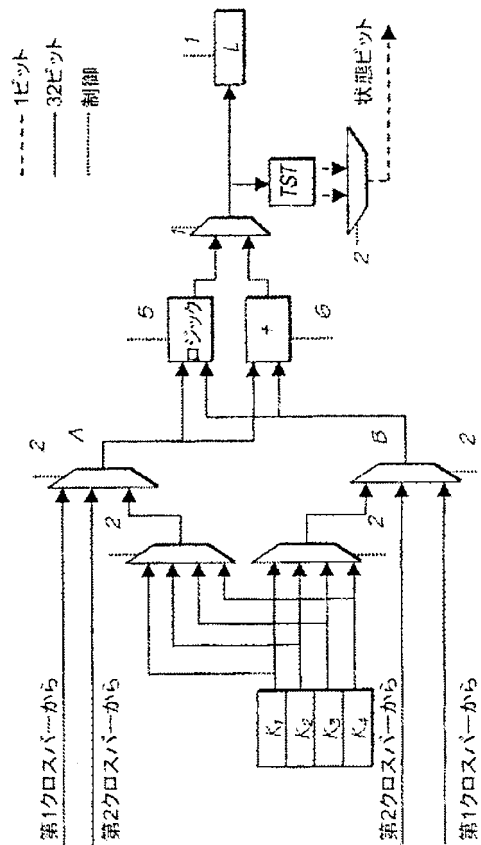
【図 1 2】



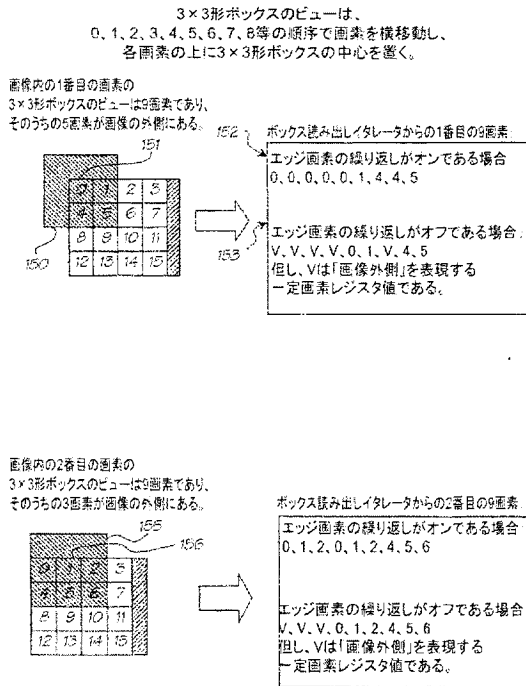
【図 1 3】



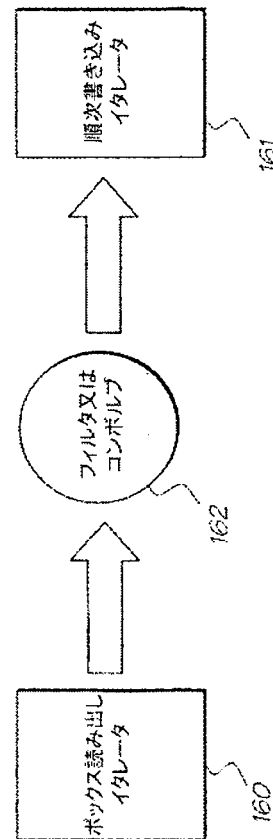
【図 1 4】



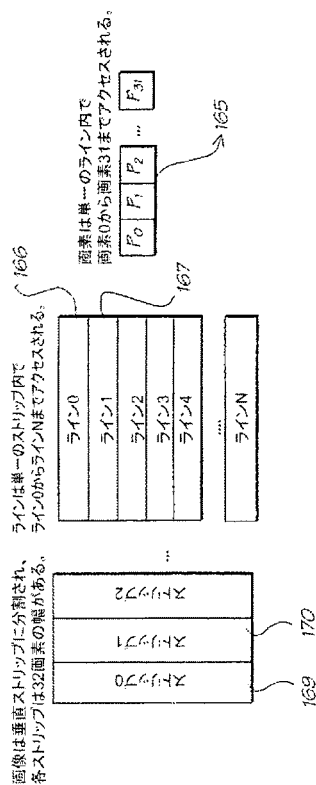
【図 19】



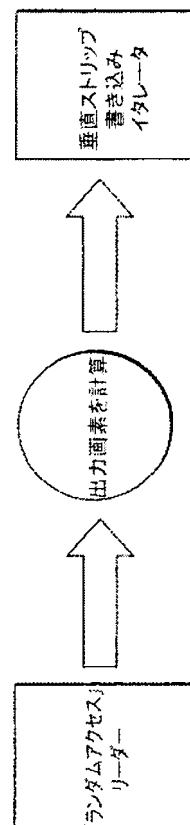
【図 20】



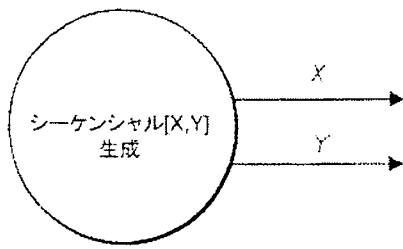
【図 21】



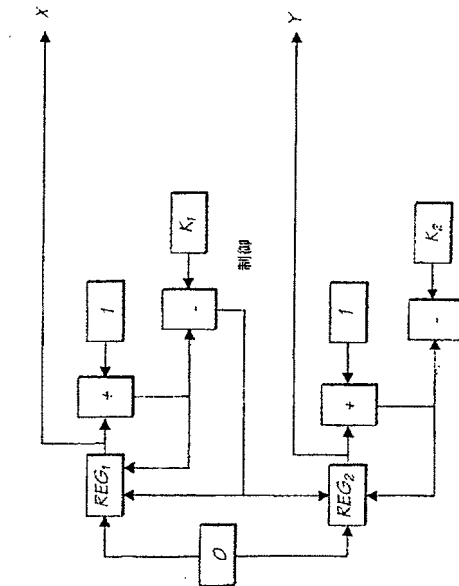
【図 22】



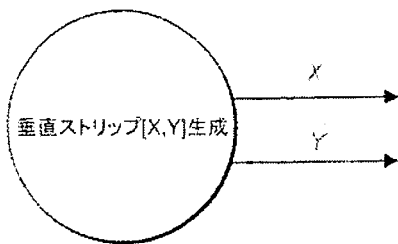
【図 2 3】



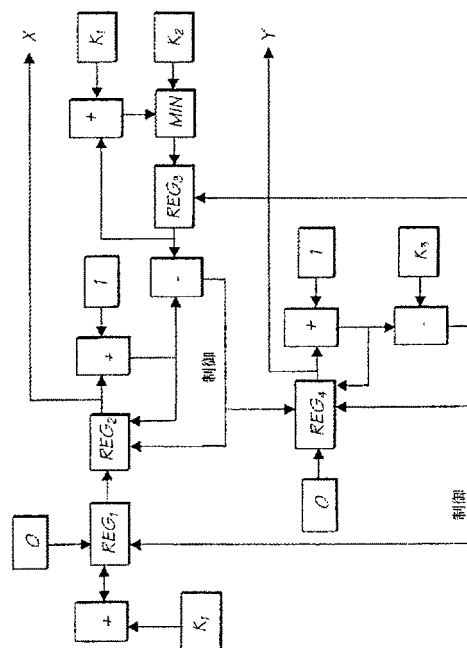
【図 2 4】



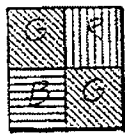
【図 2 5】



【図 2 6】

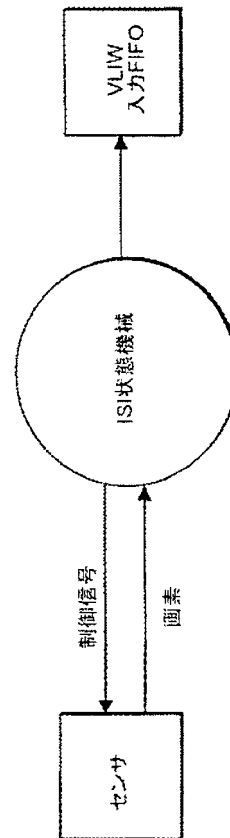


【図 27】

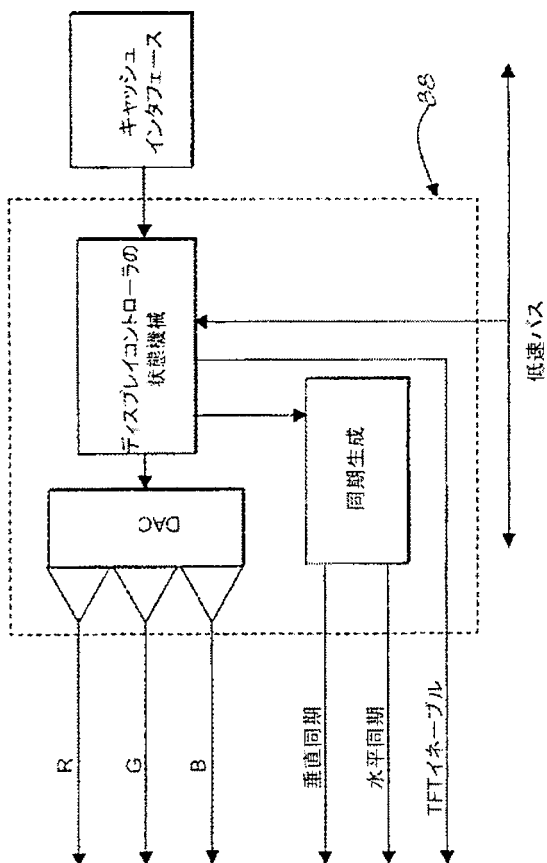


センサからの2×2画素ブロック

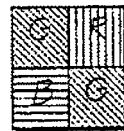
【図 28】



【図 29】

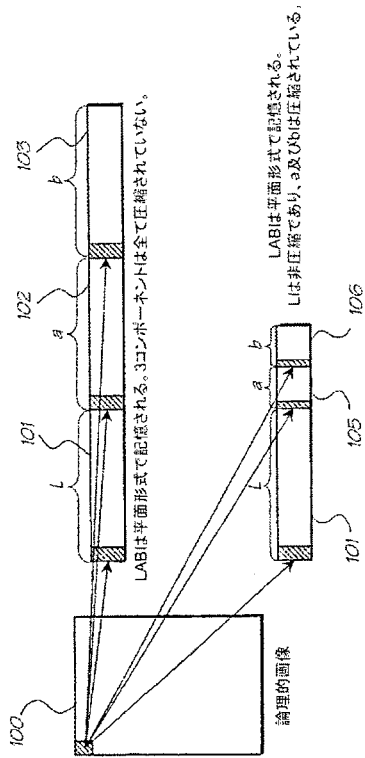


【図 30】

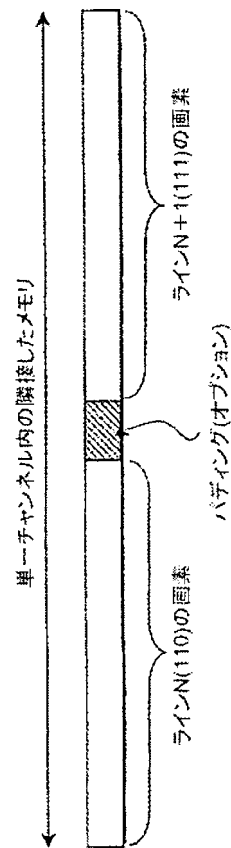


CCDからの2×2画素ブロック

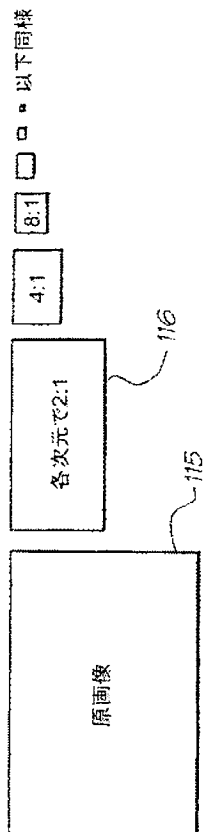
【図 3 1】



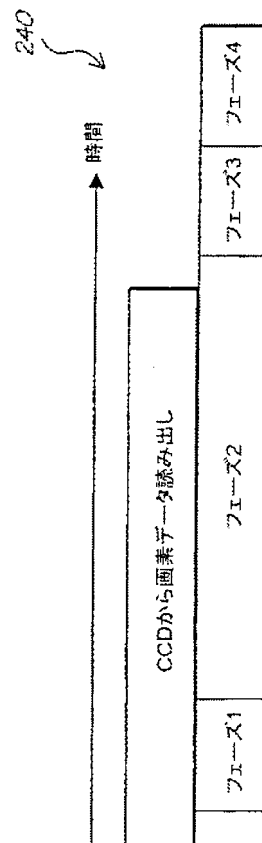
【図 3 2】



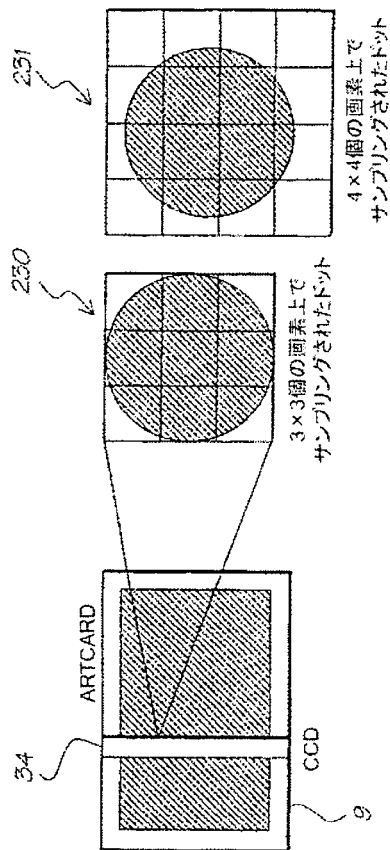
【図 3 3】



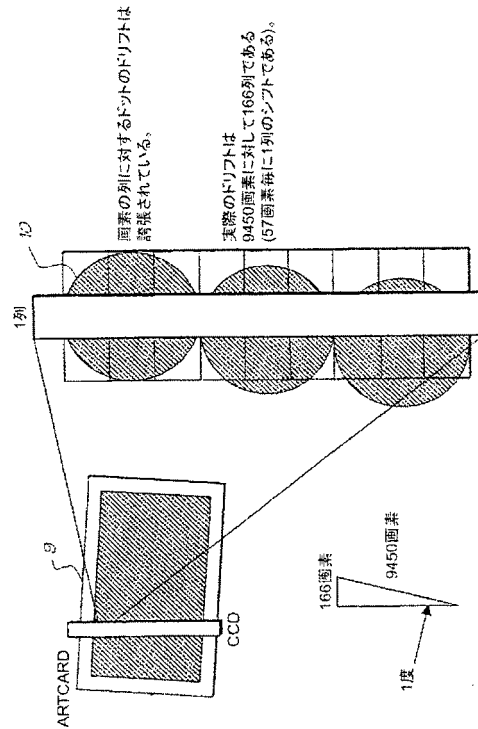
【図 3 4】



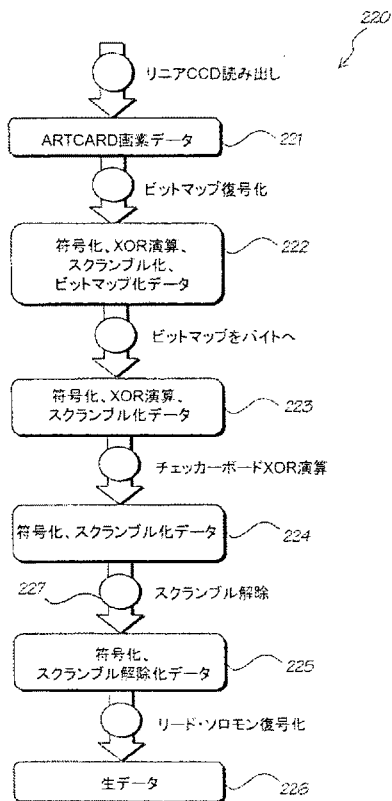
【図 35】



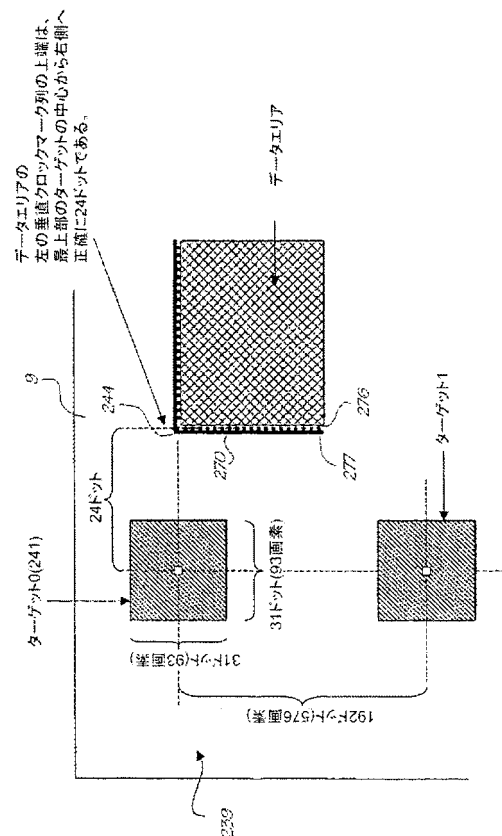
【図 36】



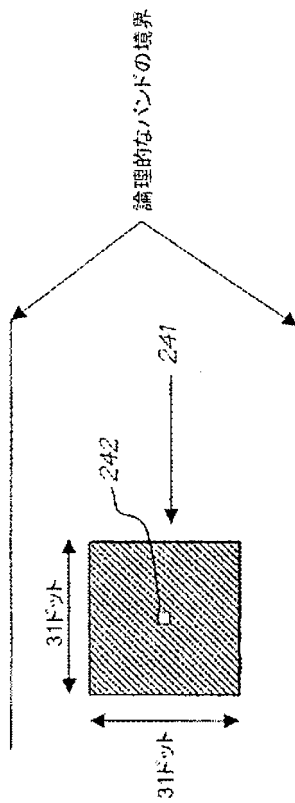
【図 37】



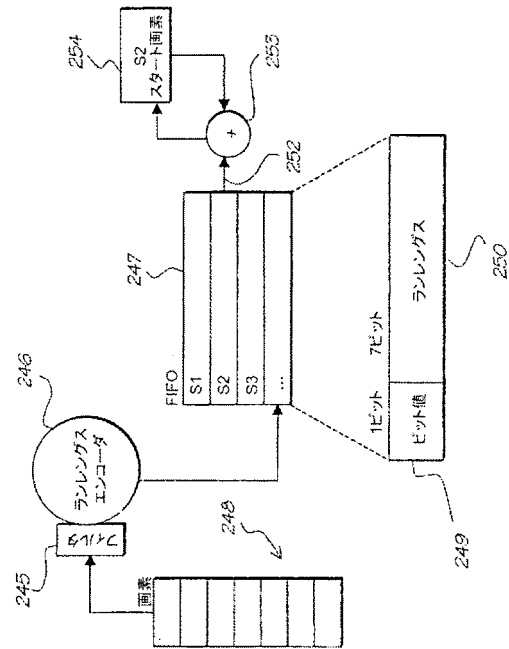
【図 38】



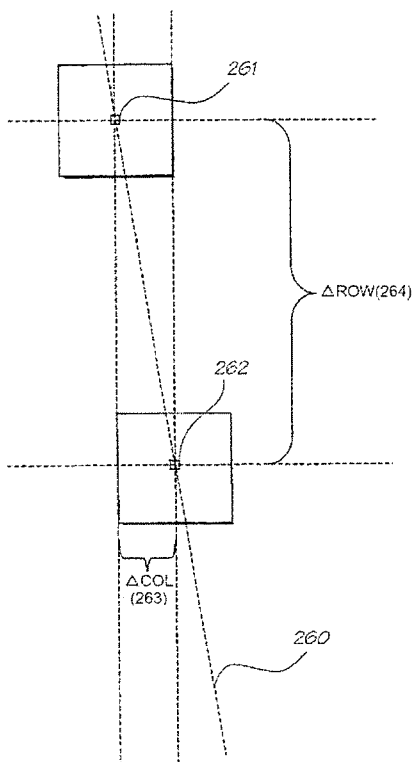
【図 39】



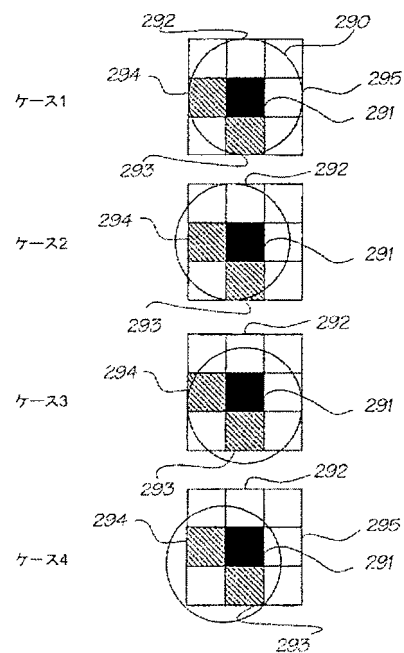
【図 40】



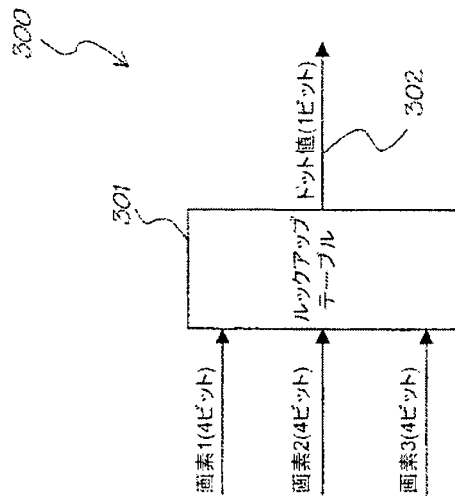
【図 41】



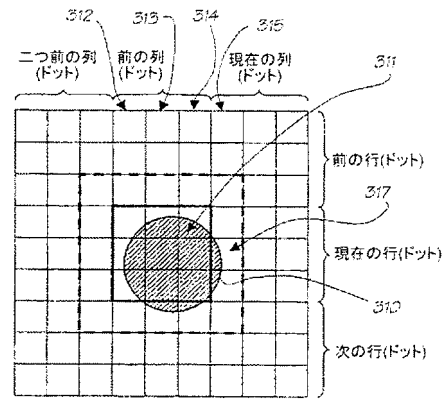
【図 42】



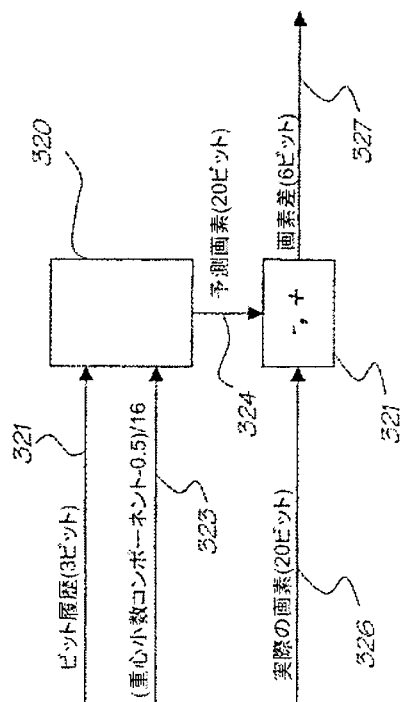
【図 4 3】



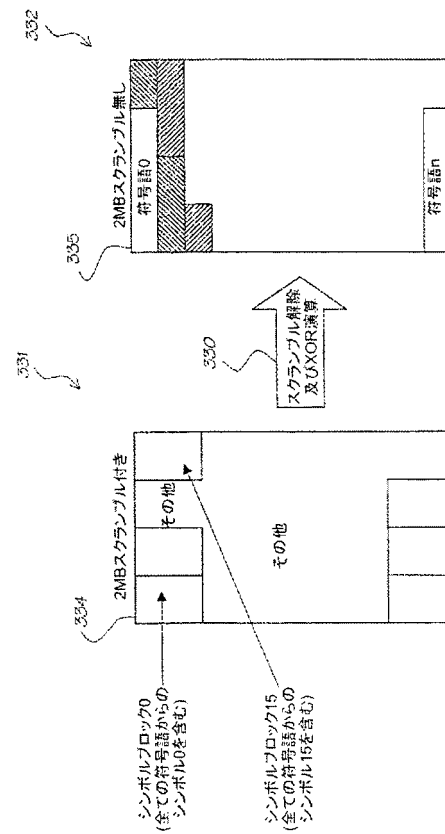
【図 4 4】



【図 4 5】



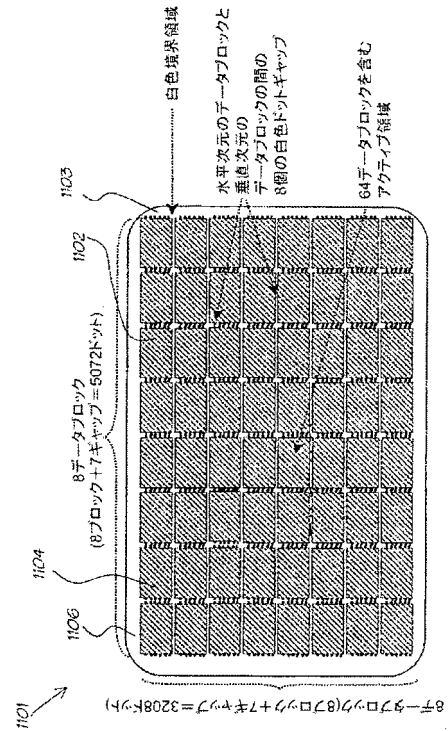
【図 4 6】



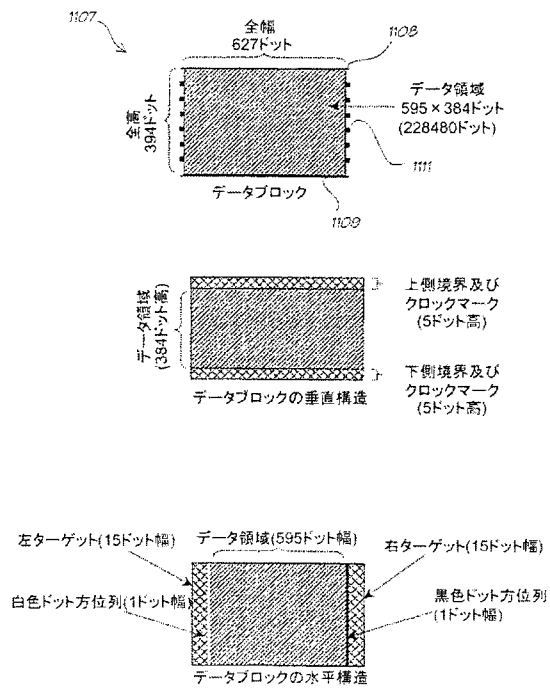
【図 47】



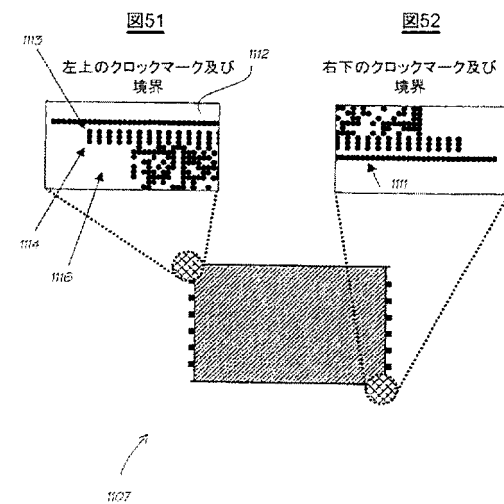
【図 48】



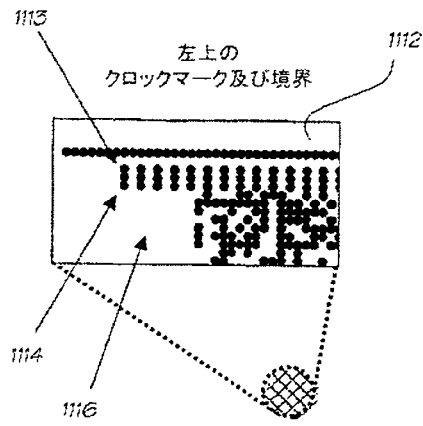
【図 49】



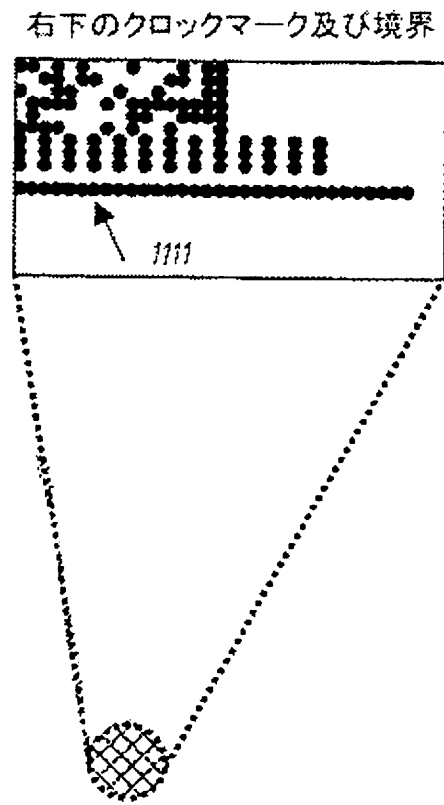
【図 50】



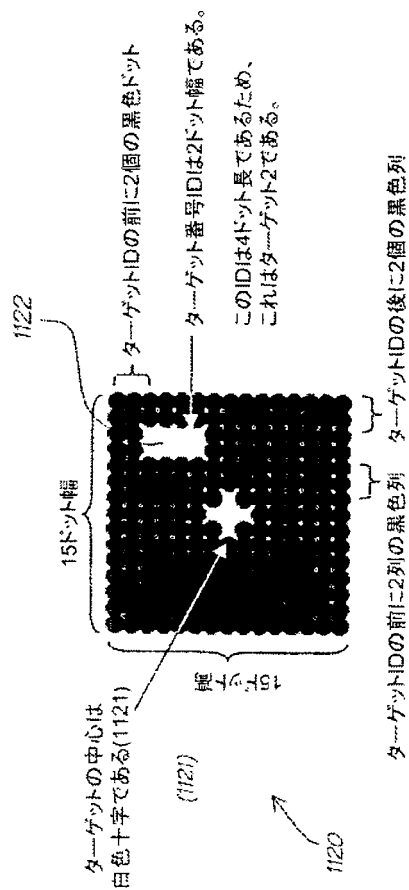
【図 5 1】



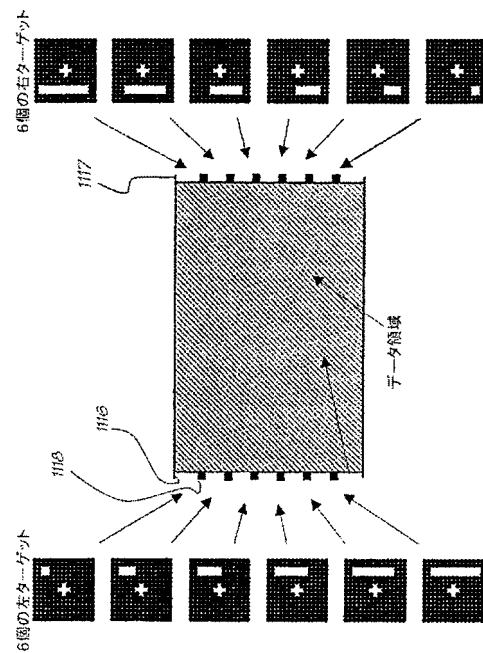
【図 5 2】



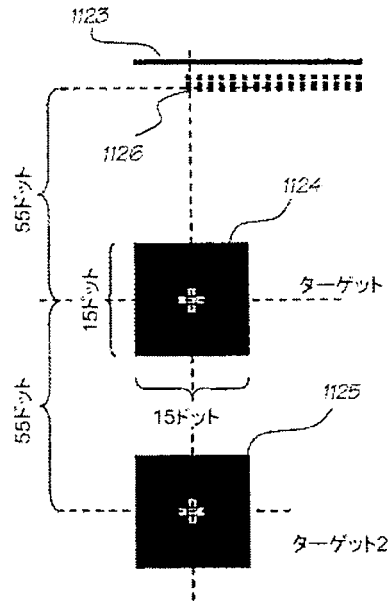
【図 5 3】



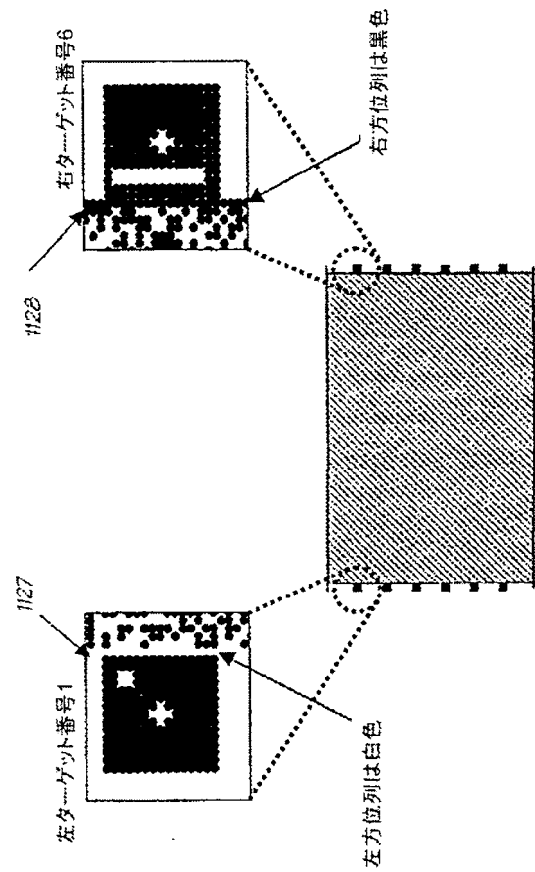
【図 5 4】



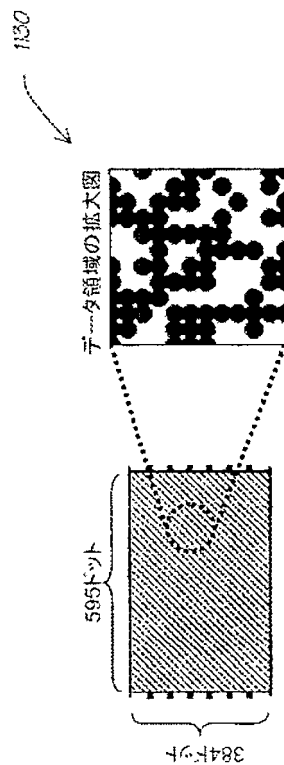
【図 5 5】



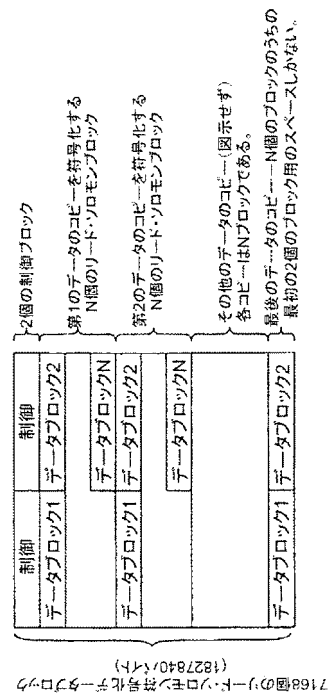
【図 5 6】



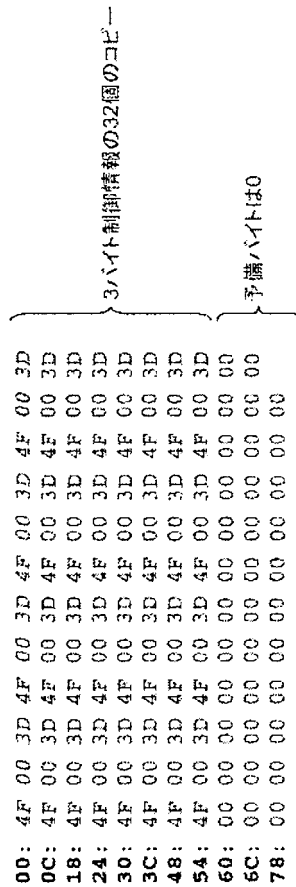
【図 5 7】



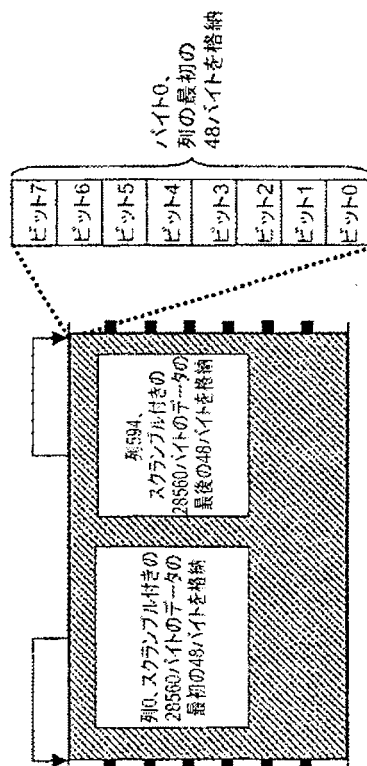
【図 5 8】



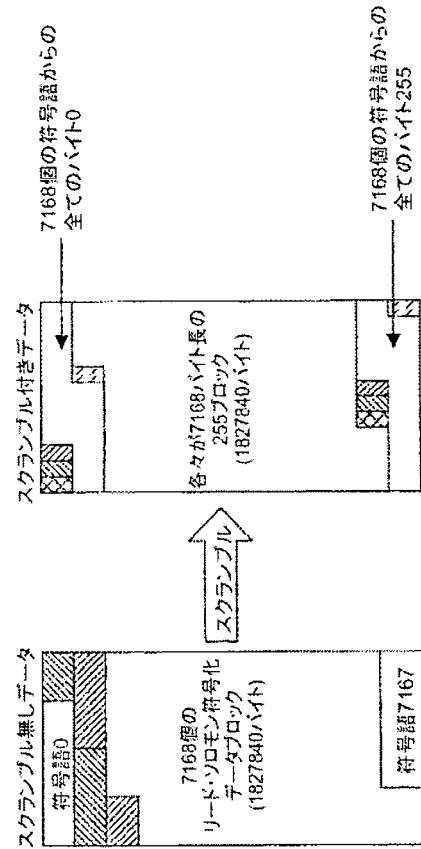
【図59】



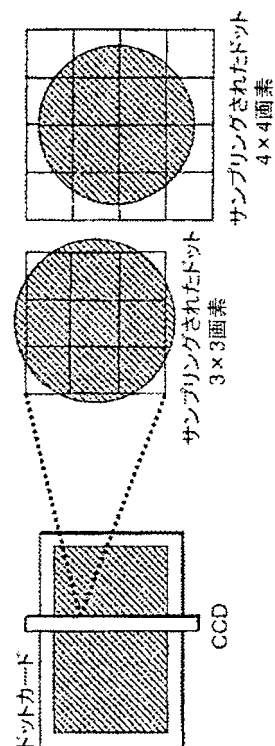
【図61】



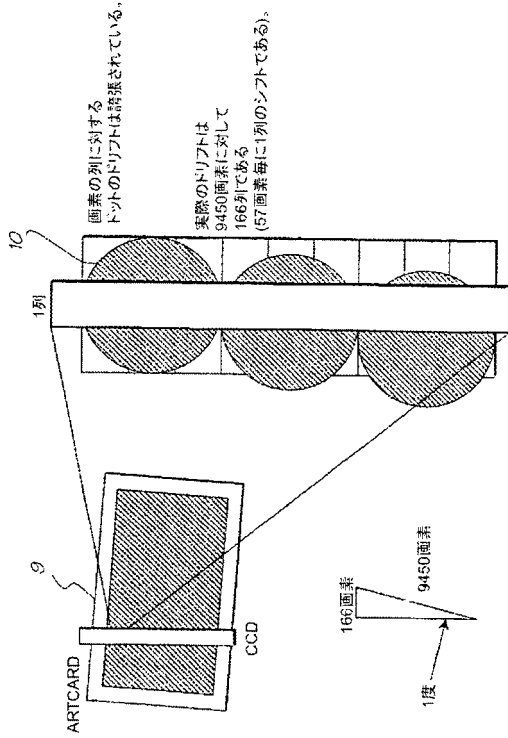
【図60】



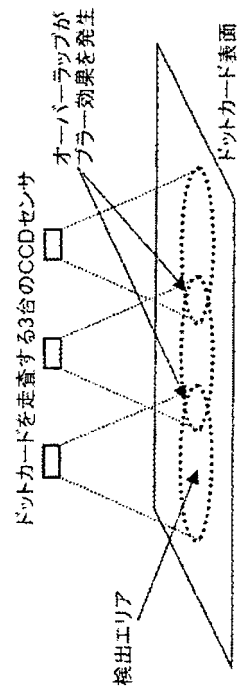
【図62】



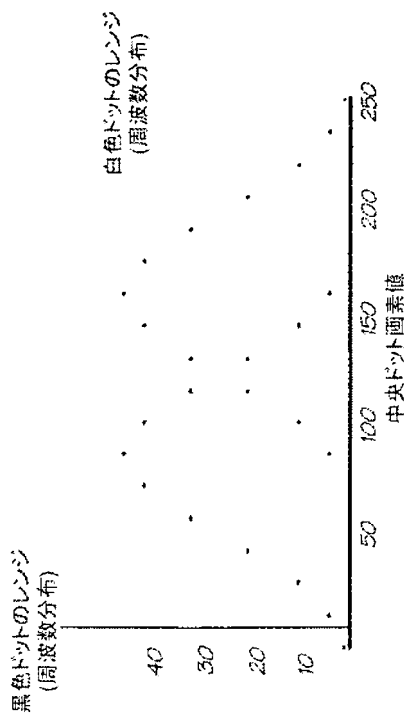
【図 6 3】



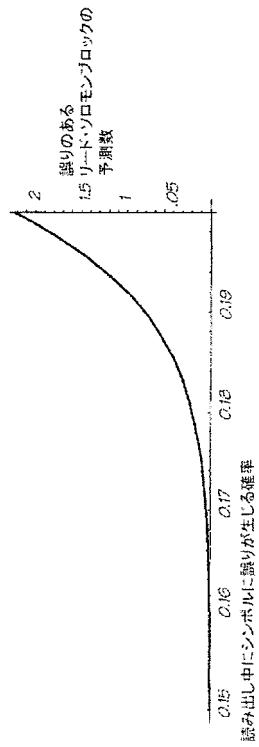
【図 6 4】



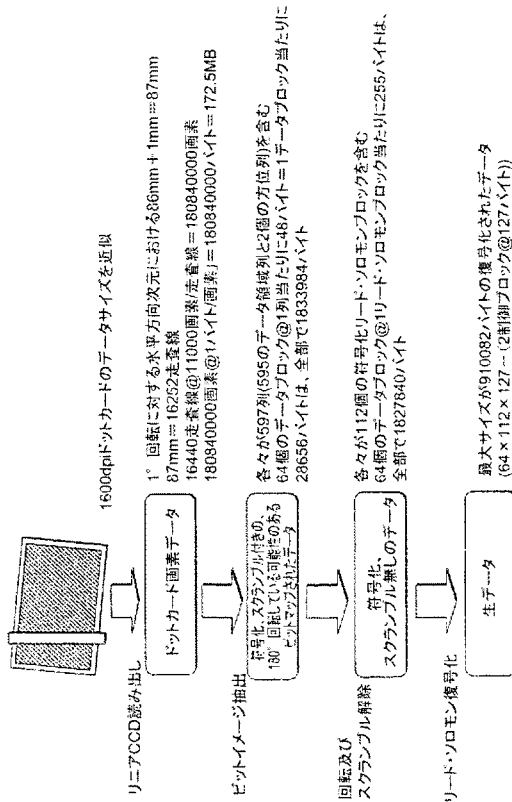
【図 6 5】



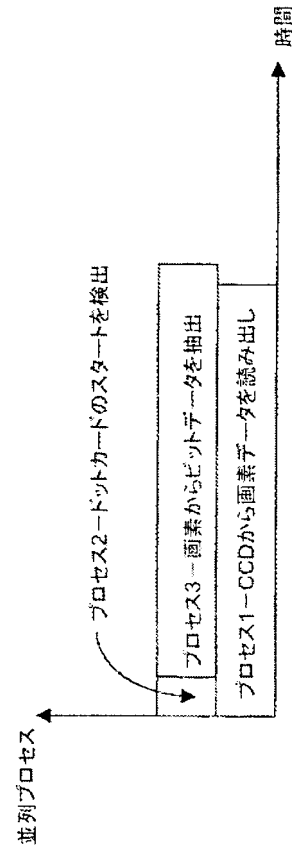
【図 6 6】



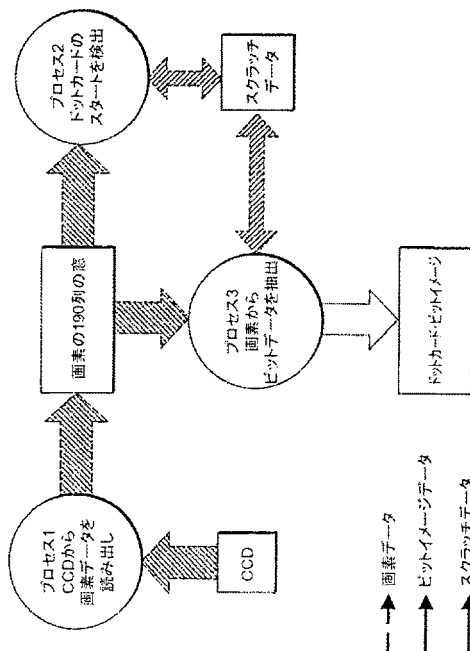
【図 67】



【図 68】



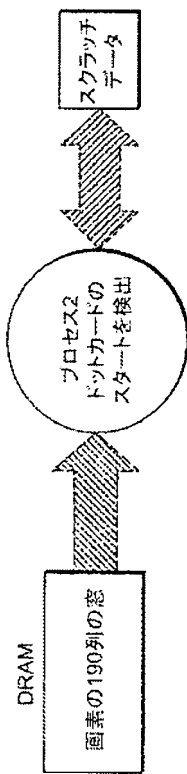
【図 69】



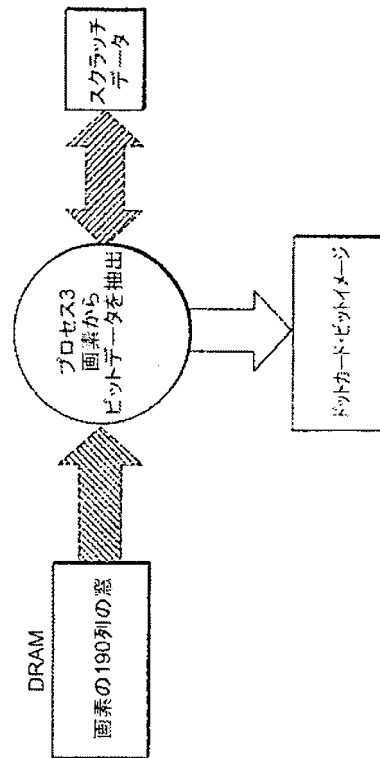
【図 70】



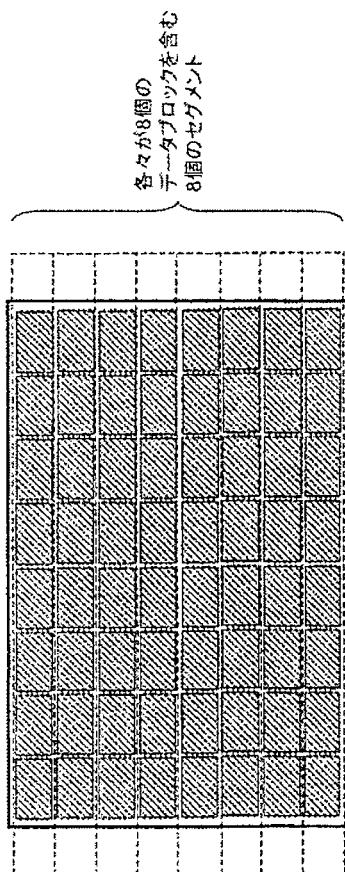
【図 7 1】



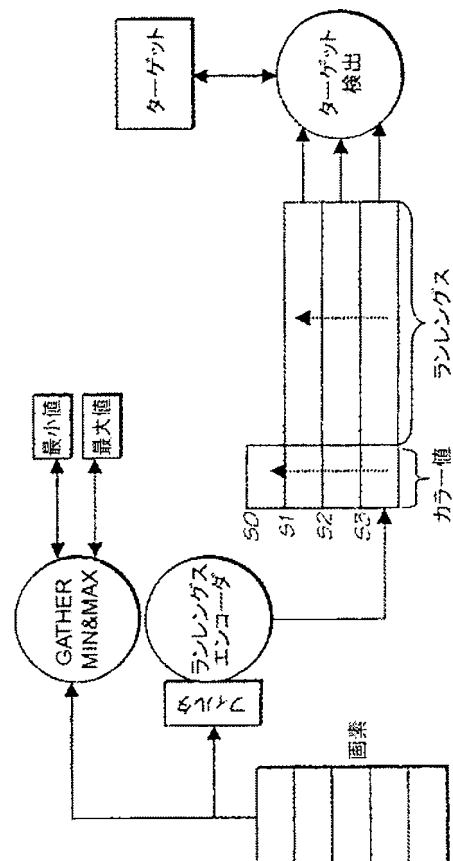
【図 7 2】



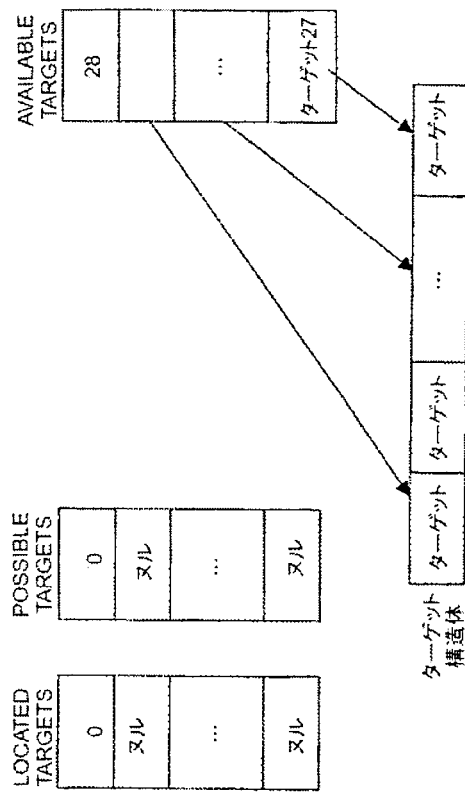
【図 7 3】



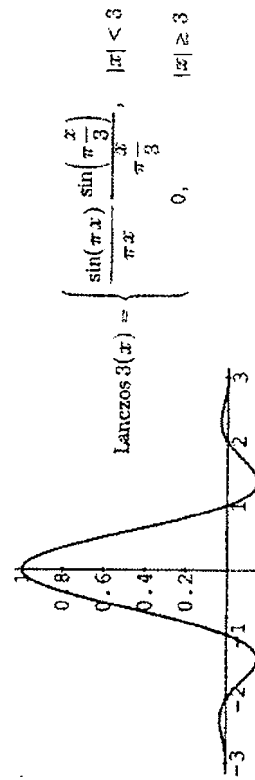
【図 7 4】



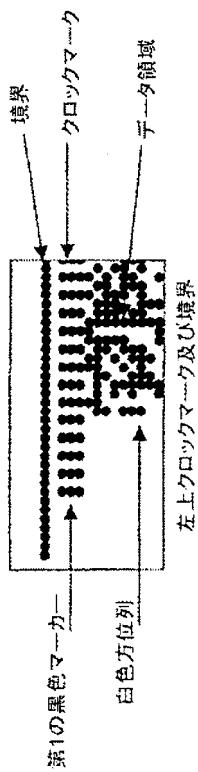
【図 75】



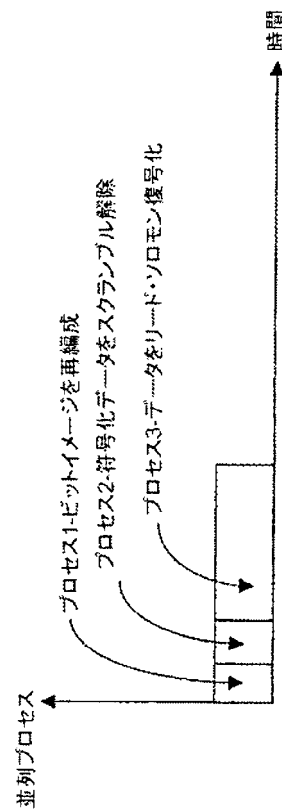
【図 76】



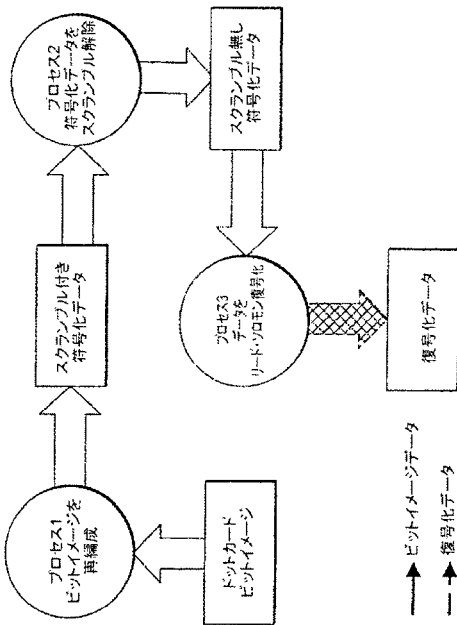
【図 77】



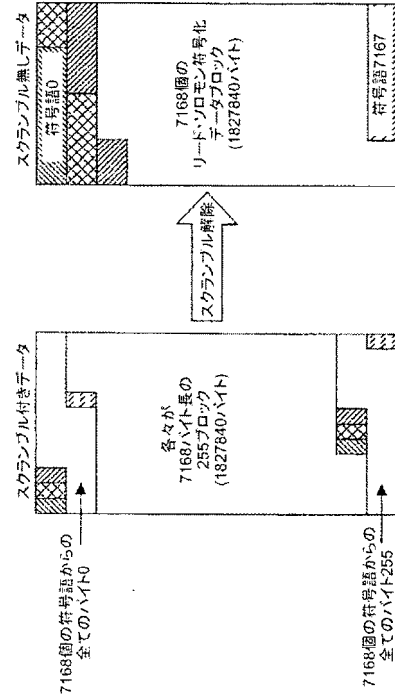
【図 78】



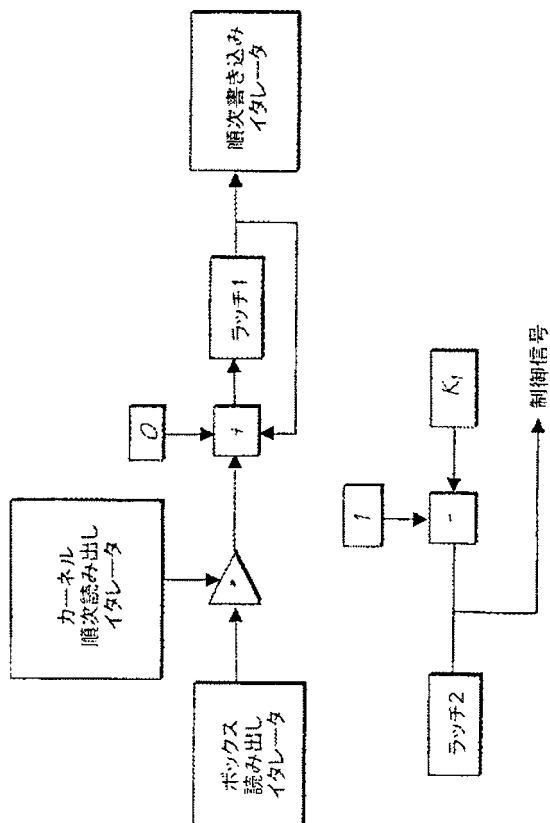
【図 79】



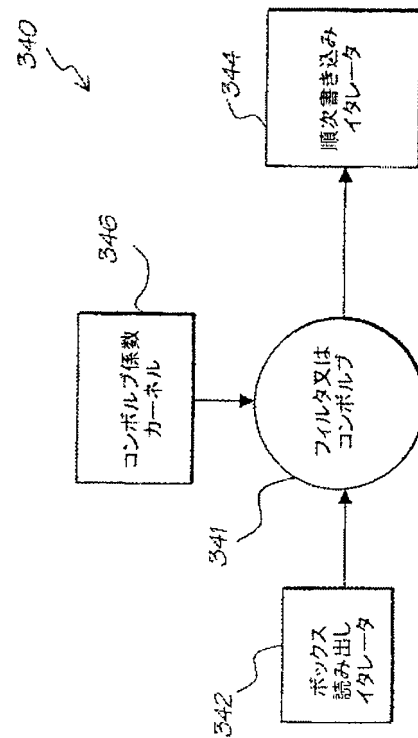
【図 80】



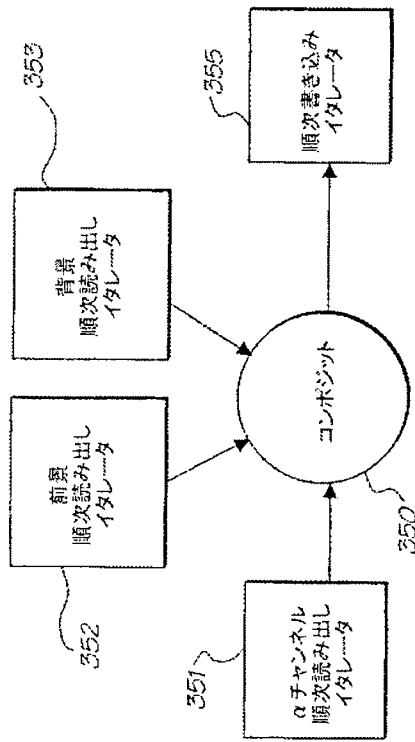
【図 81】



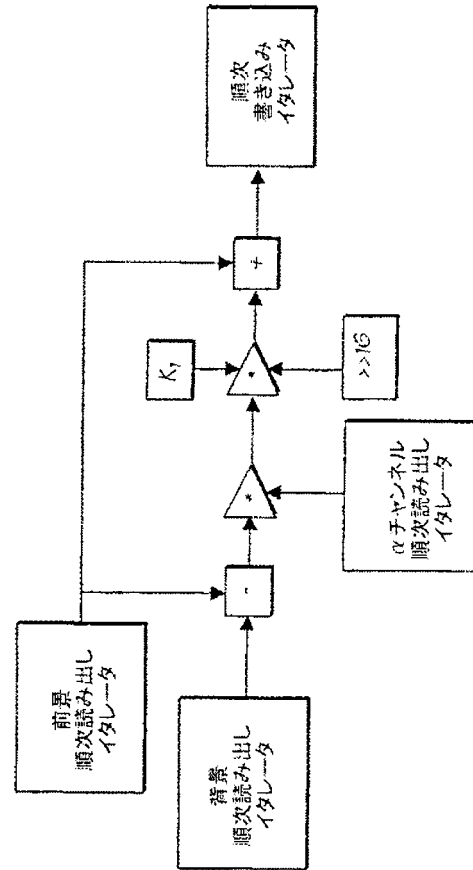
【図 82】



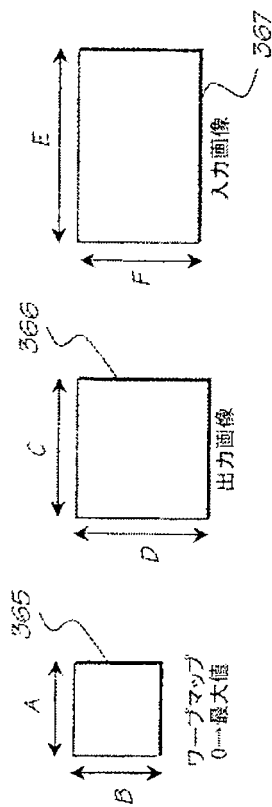
【図 8 3】



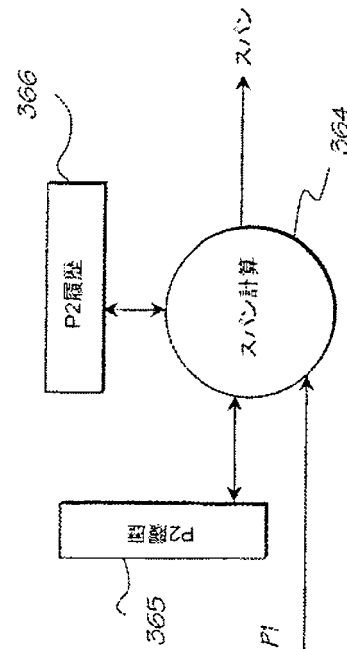
【図 8 4】



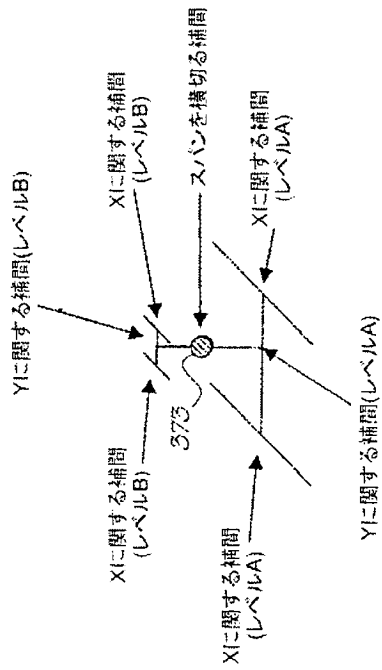
【図 8 5】



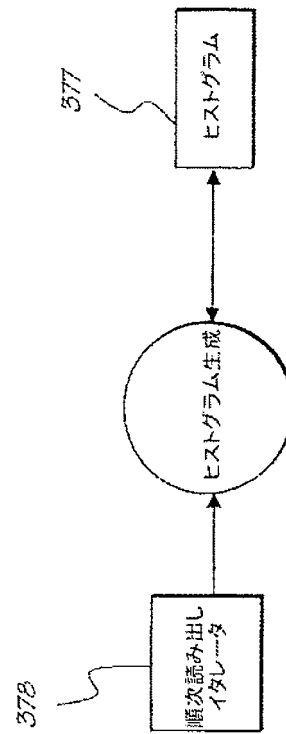
【図 8 6】



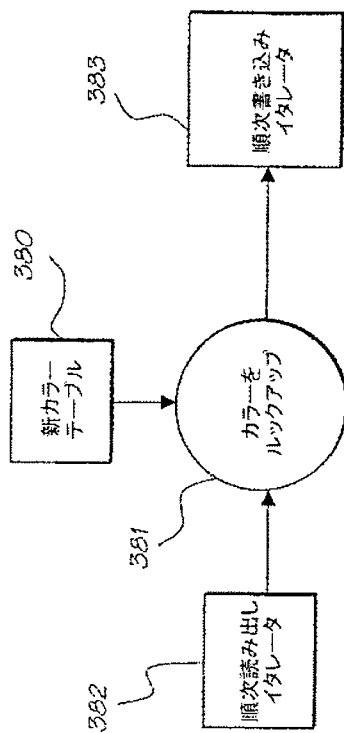
【図 9 1】



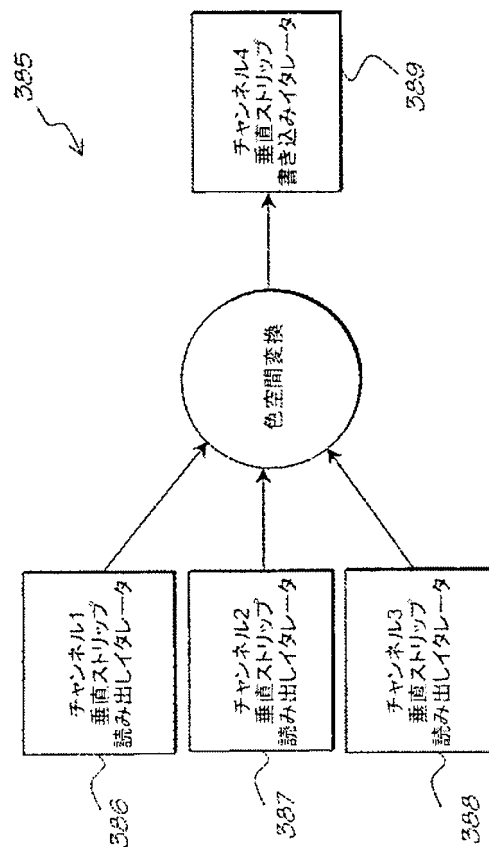
【図 9 2】



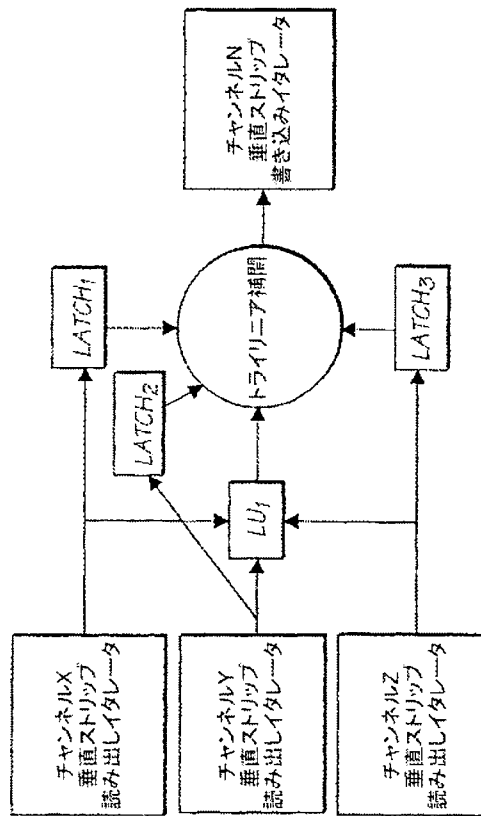
【図 9 3】



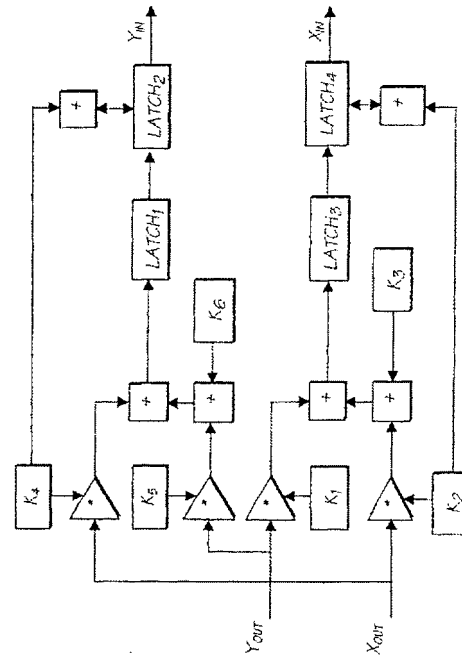
【図 9 4】



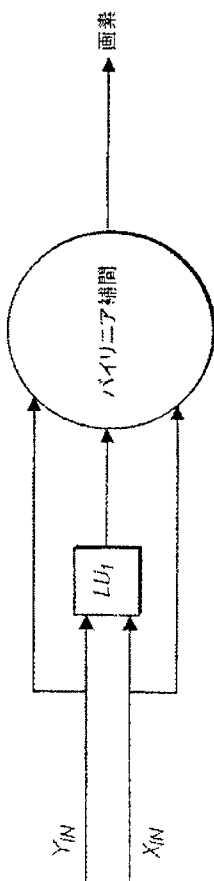
【図 95】



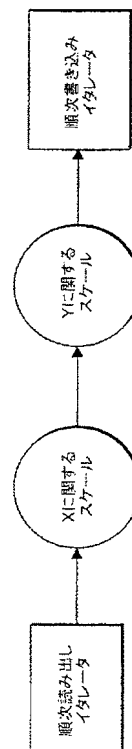
【図 96】



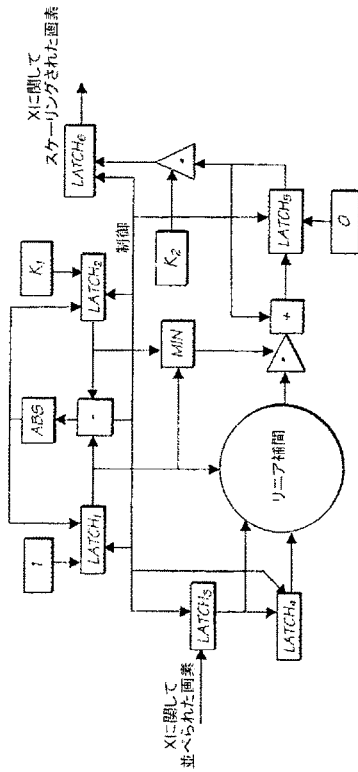
【図 97】



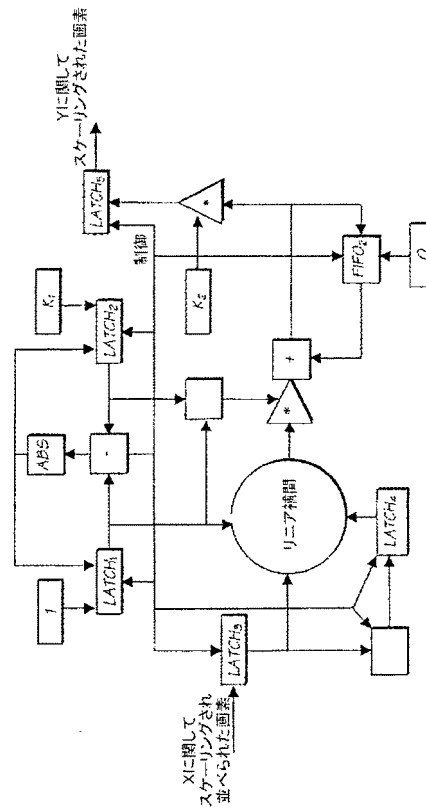
【図 98】



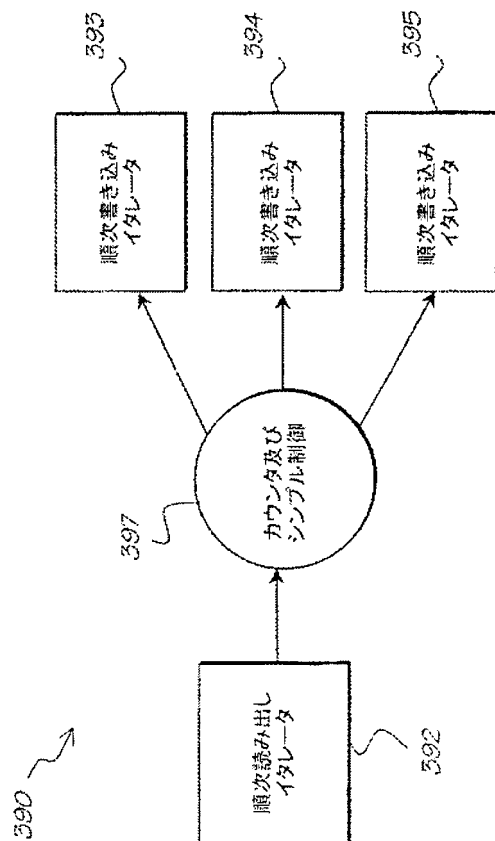
【図 99】



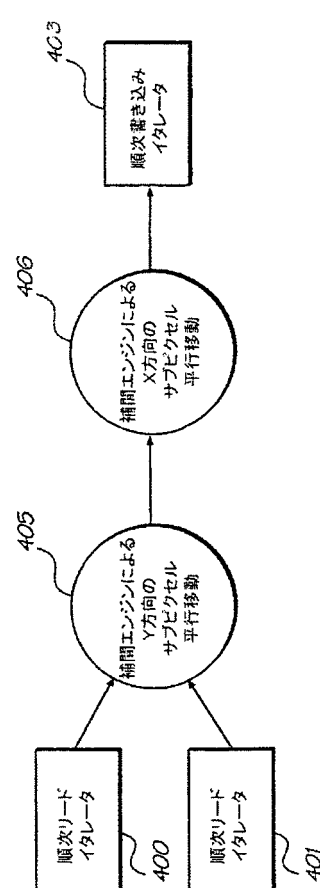
【図 100】



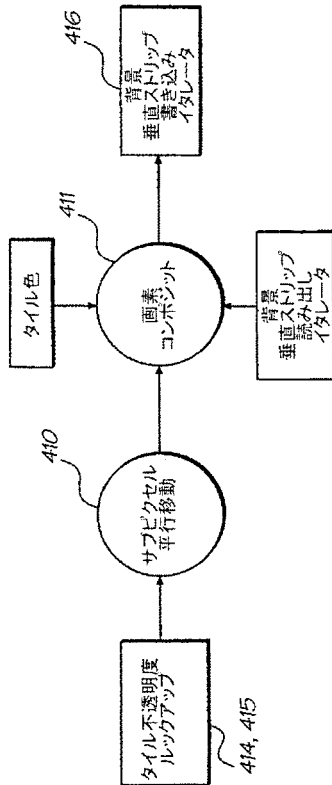
【図 101】



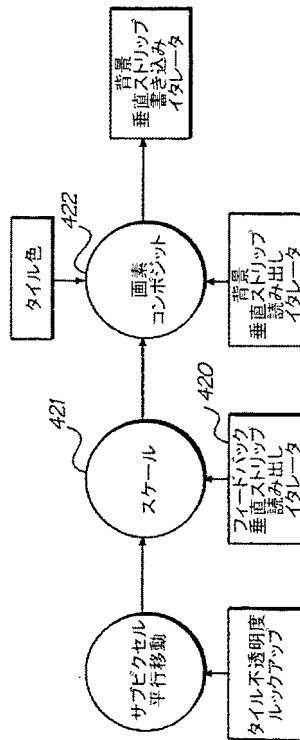
【図 102】



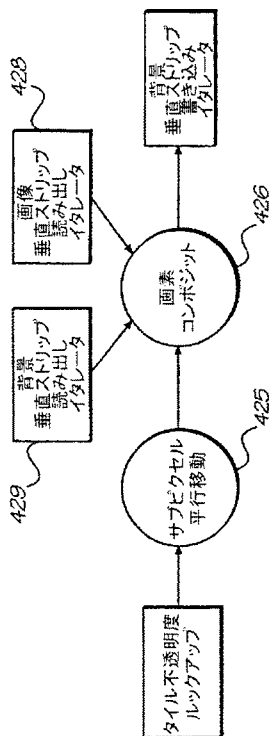
【図 103】



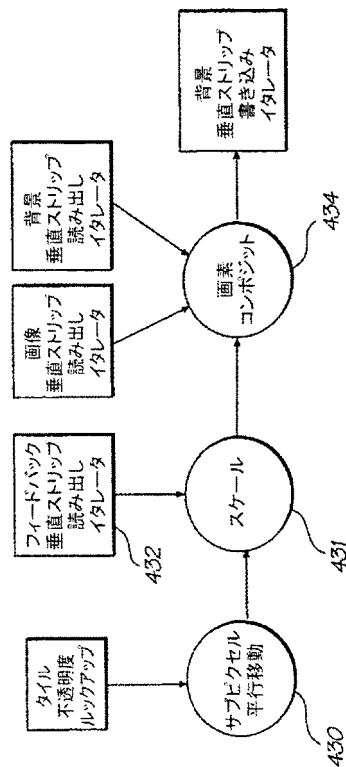
【図 104】



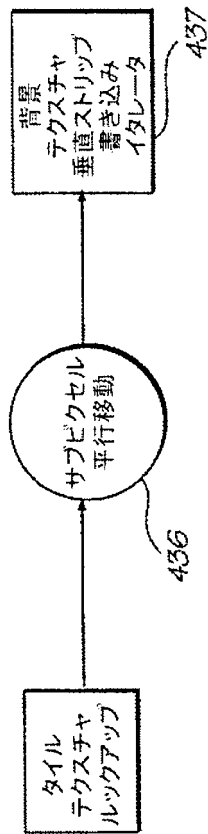
【図 105】



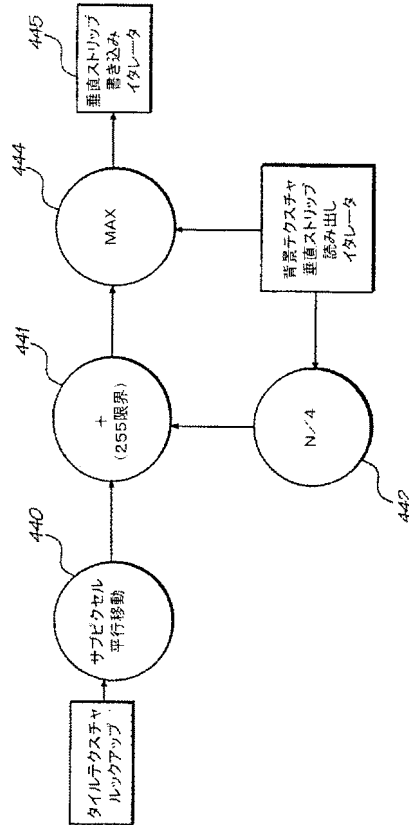
【図 106】



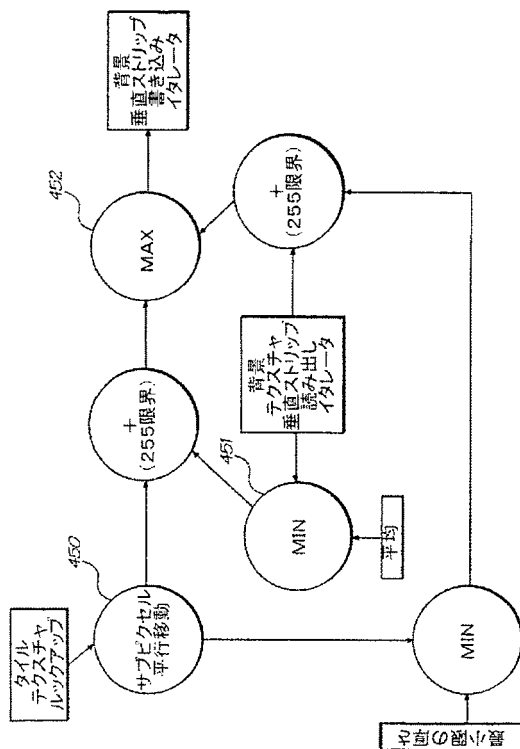
【図107】



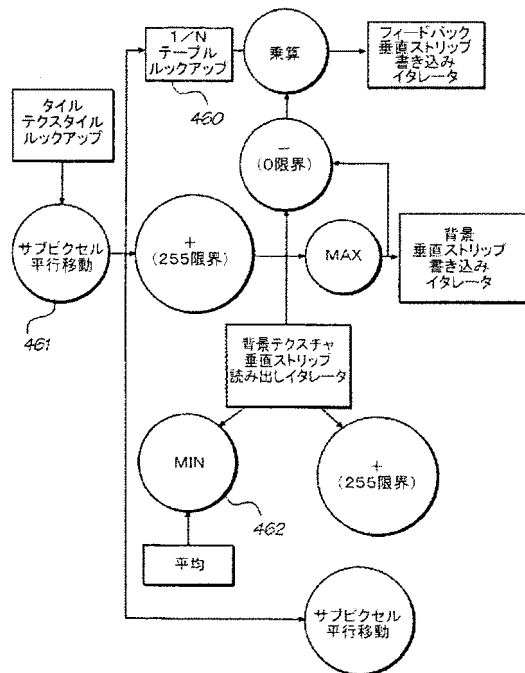
【図108】



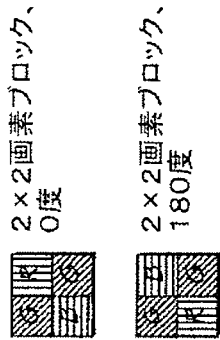
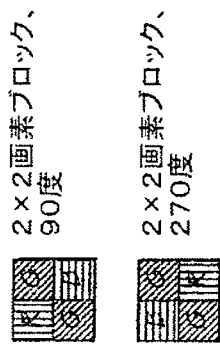
【図109】



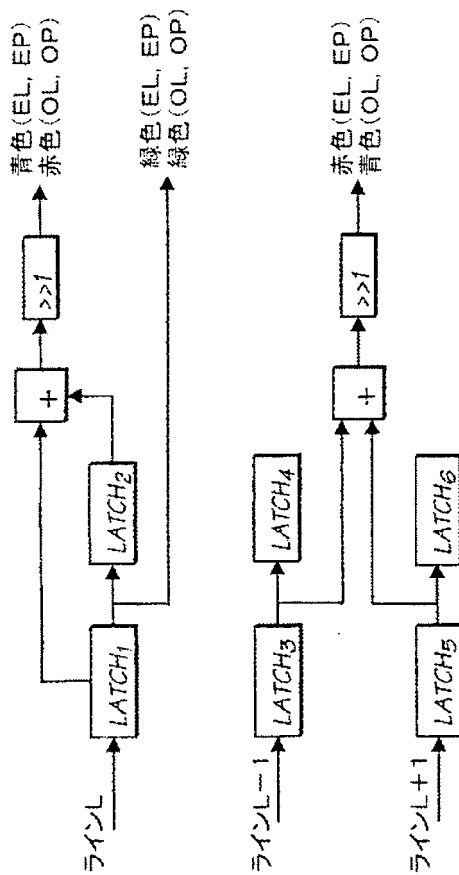
【図110】



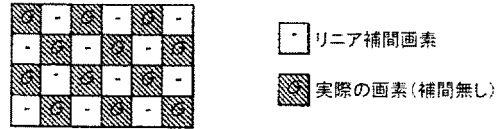
【図 1 1 1】



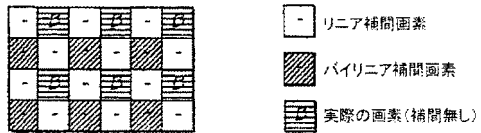
【図 1 1 5】



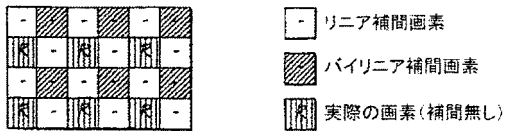
【図 1 1 2】



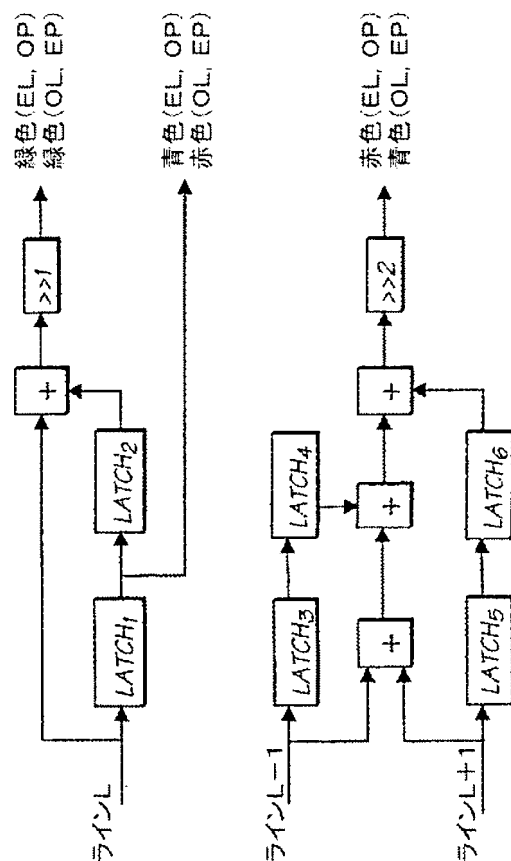
【図 1 1 3】



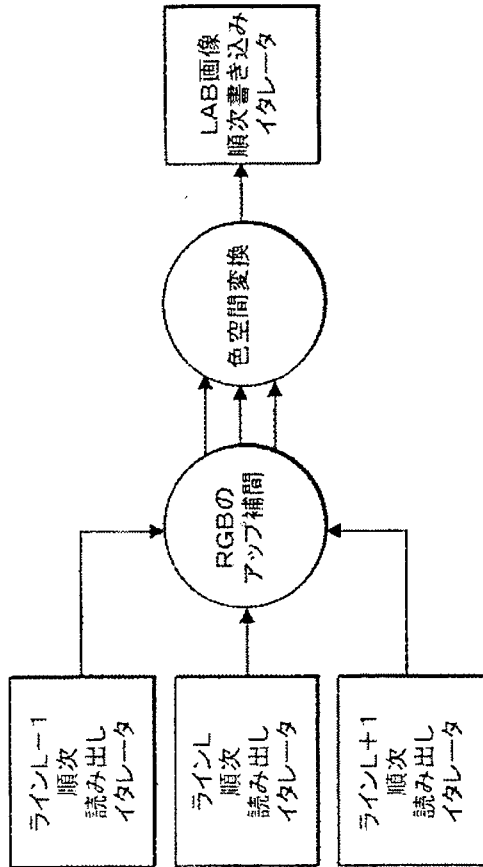
【図 1 1 4】



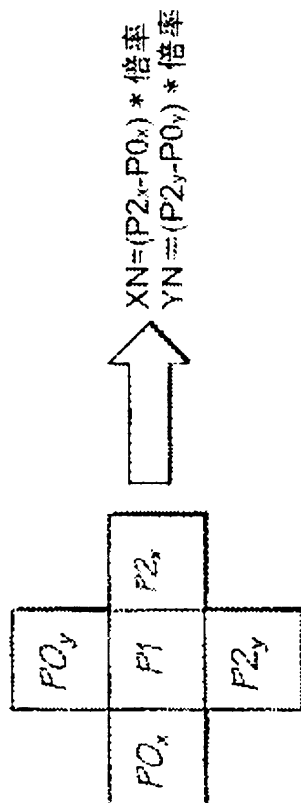
【図 1 1 6】



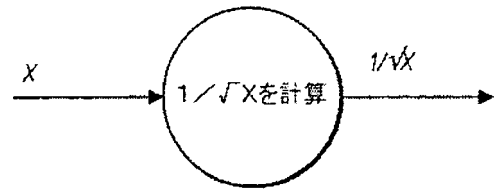
【図 1 1 7】



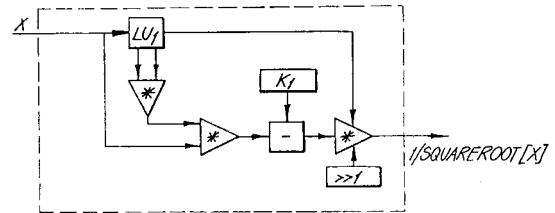
【図 1 2 0】



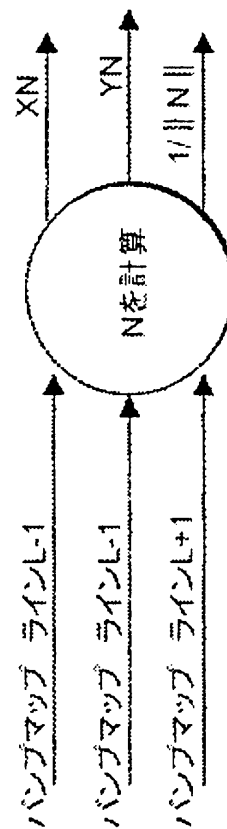
【図 1 1 8】



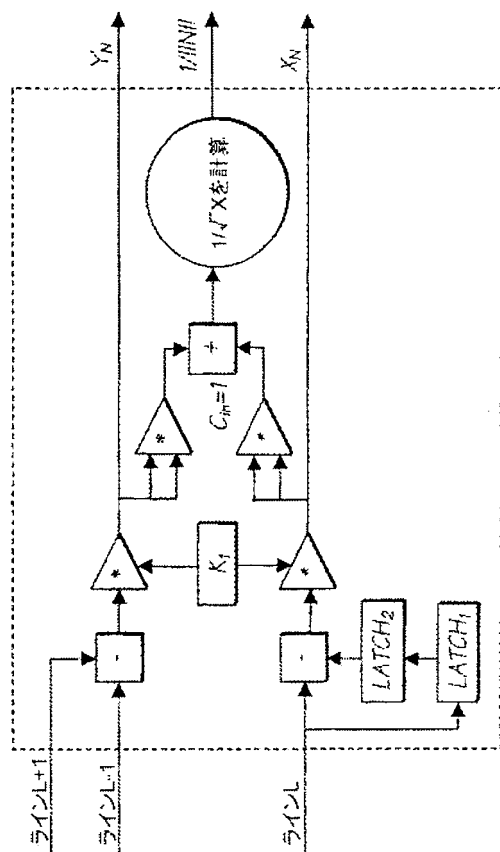
【図 1 1 9】



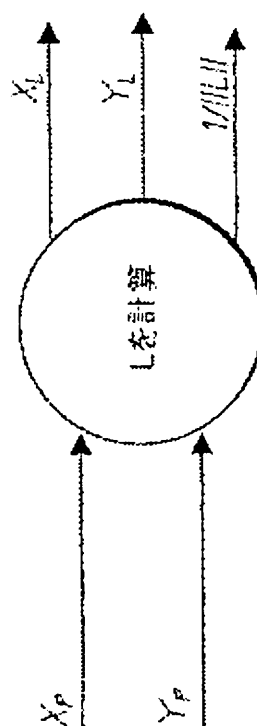
【図 1 2 1】



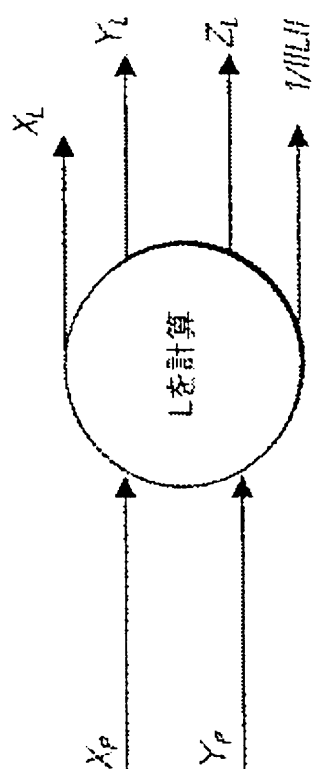
【図 1 2 2】



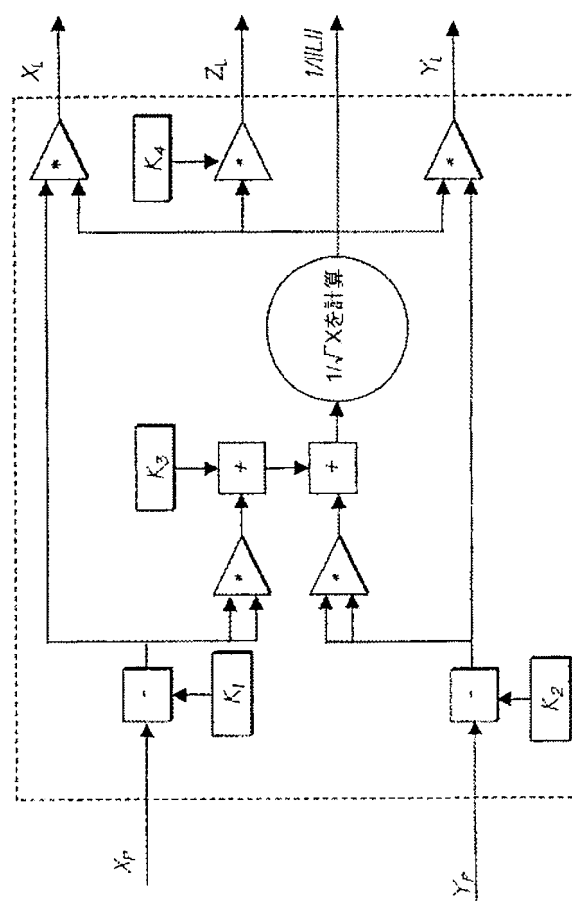
【図 1 2 3】



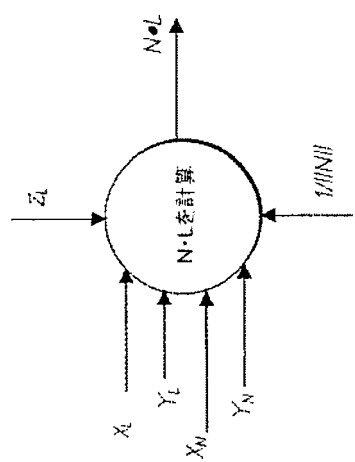
【 図 1 2 4 】



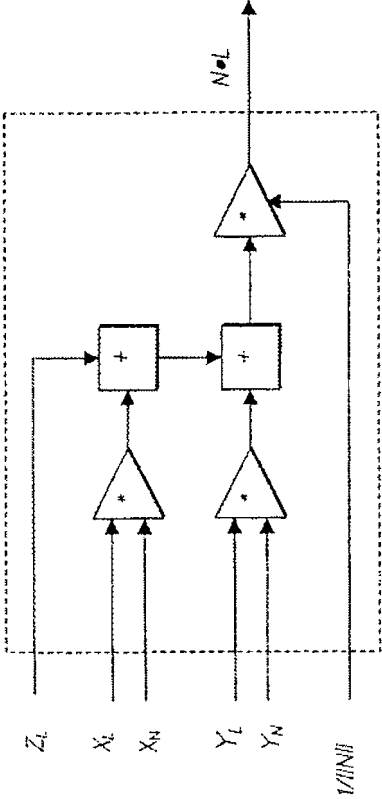
【図 1 2 5】



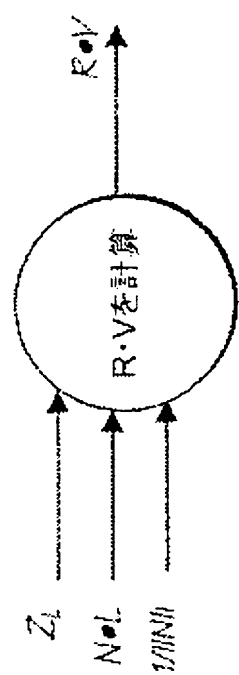
【図 1 2 6】



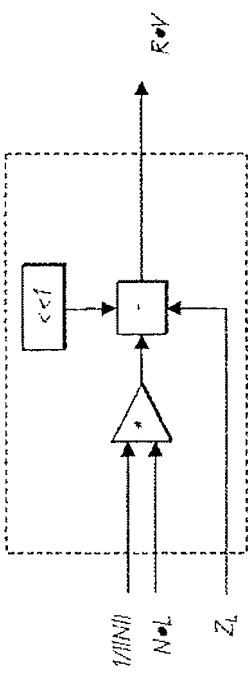
【図 1 2 7】



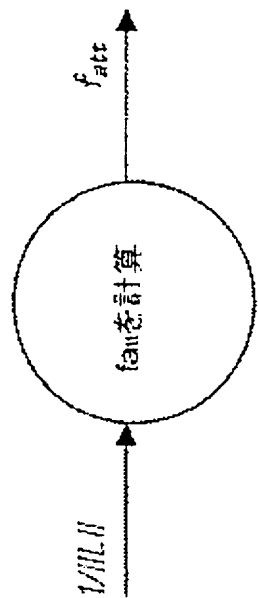
【図 1 2 8】



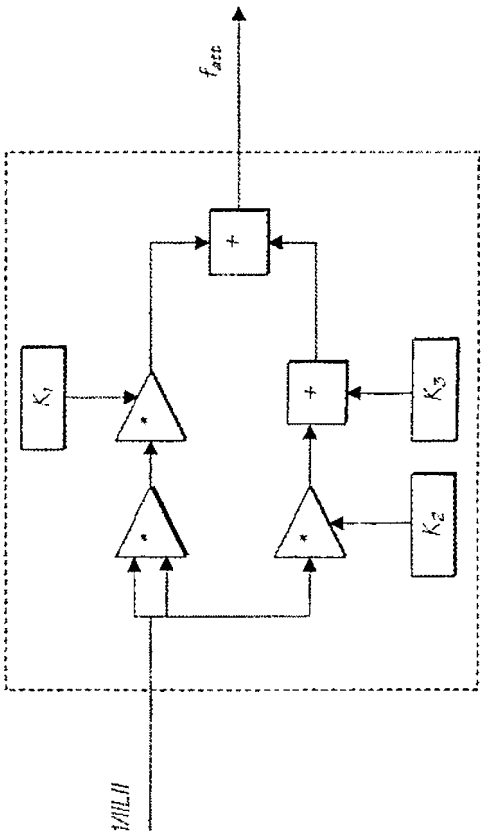
【図 1 2 9】



【図 1 3 0】



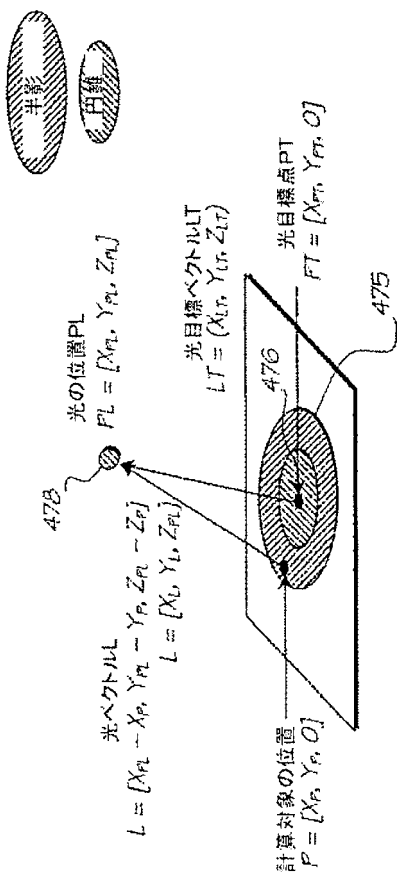
【図 1 3 1】



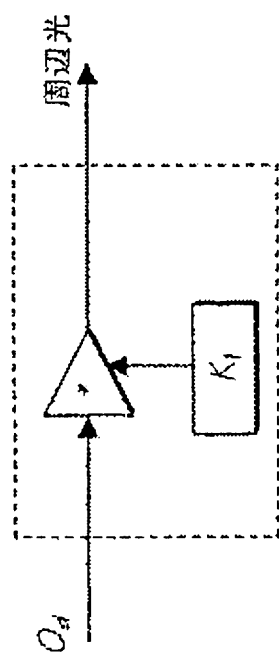
【図 1 3 2】



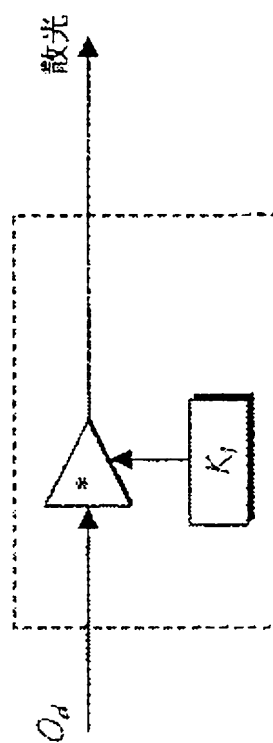
【図 1 3 3】



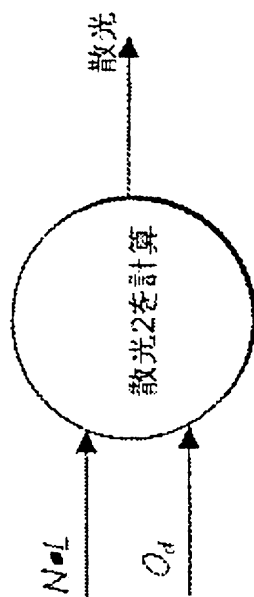
【図 1 3 8】



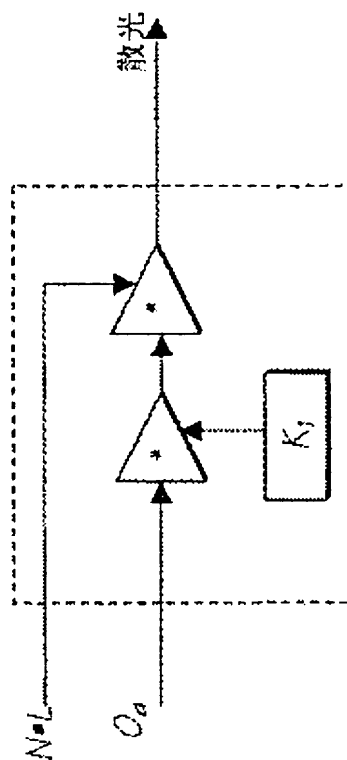
【図 1 3 9】



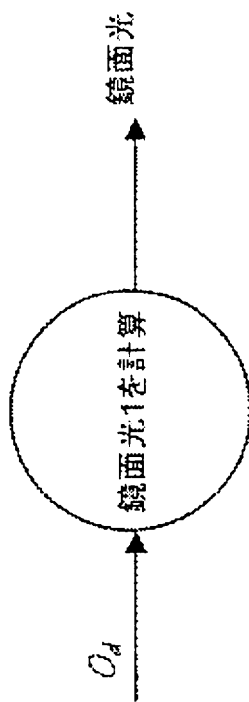
【図 1 4 0】



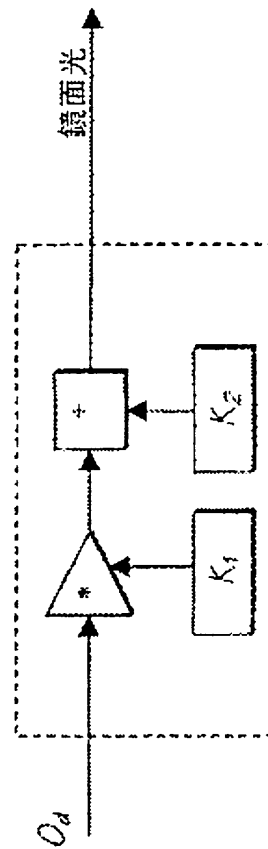
【図 1 4 1】



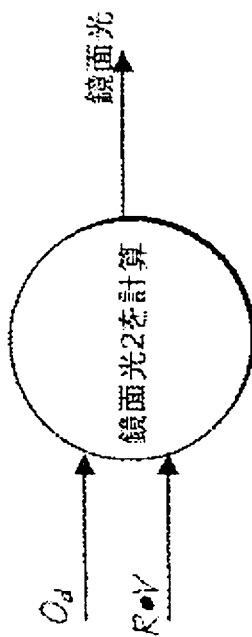
【図 1 4 2】



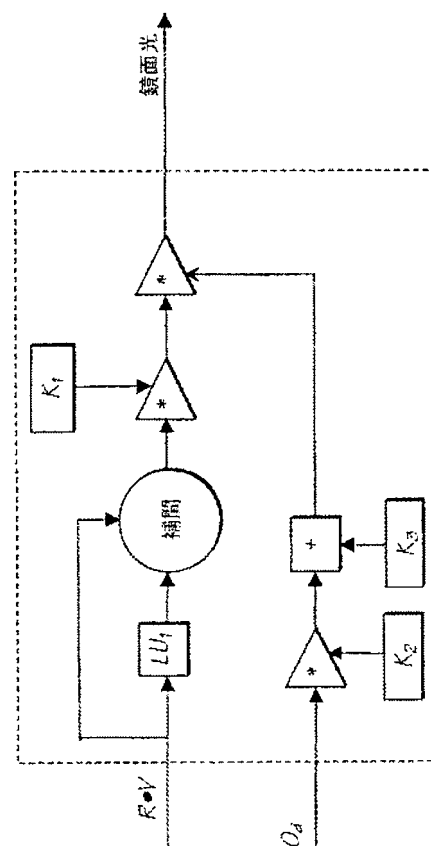
【図 1 4 3】



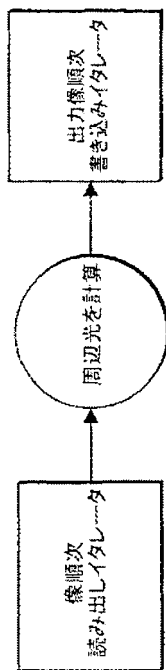
【図 1 4 4】



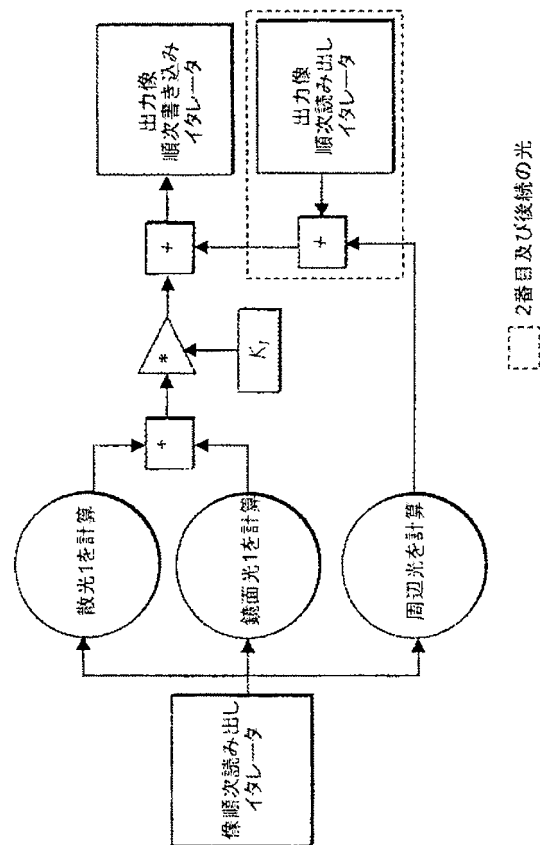
【図 1 4 5】



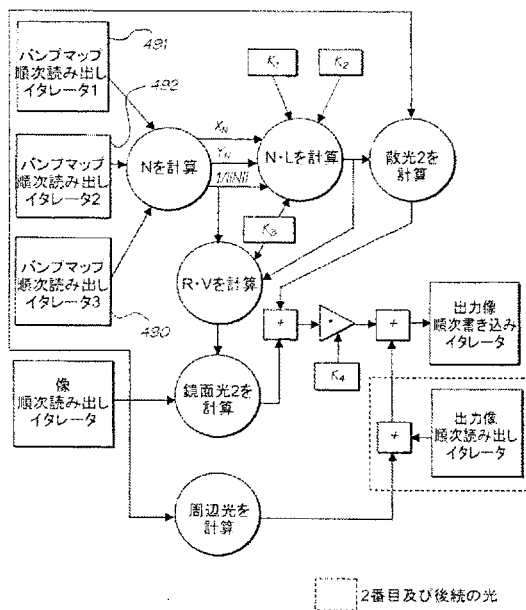
【図 1 4 6】



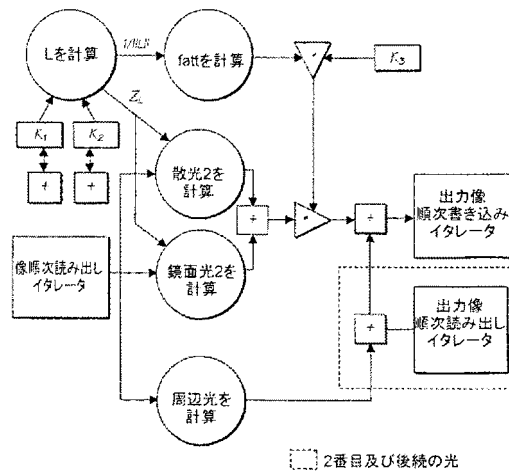
【図 1 4 7】



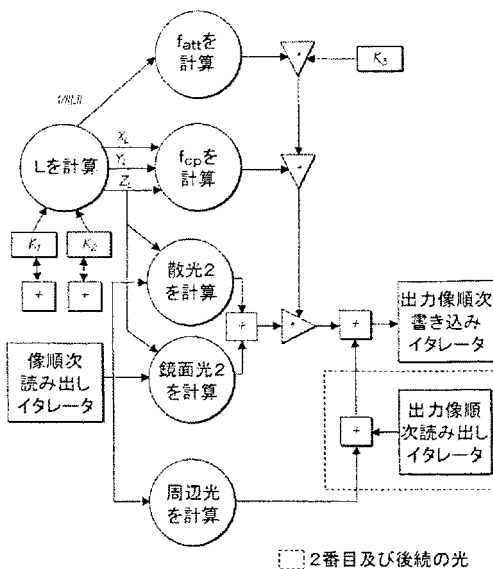
【図 1 4 8】



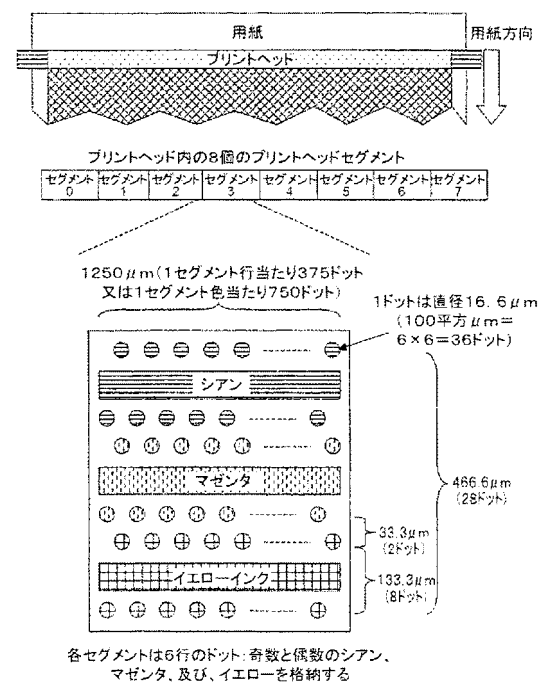
【図 1 4 9】



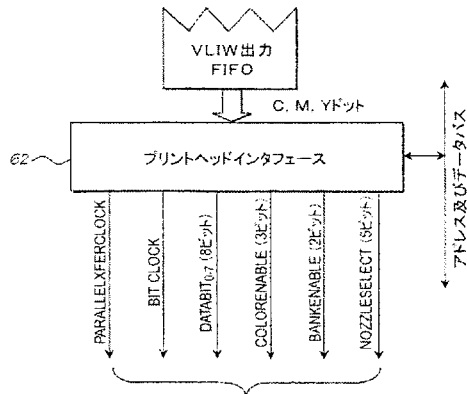
【図 15 1】



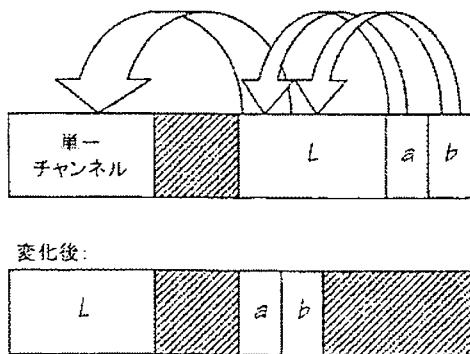
【 図 1 5 3 】



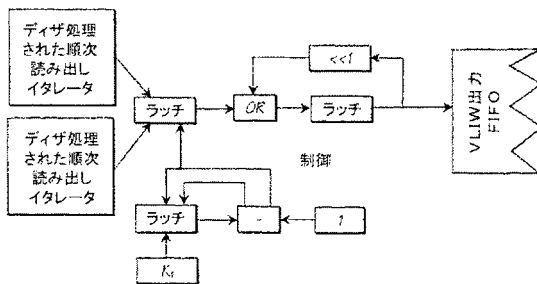
【図154】



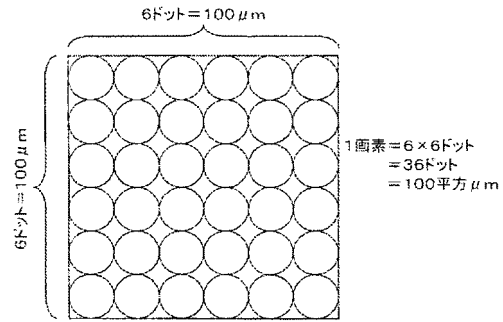
【図155】



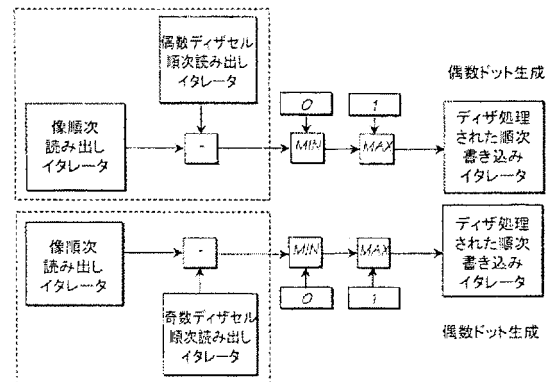
【図158】



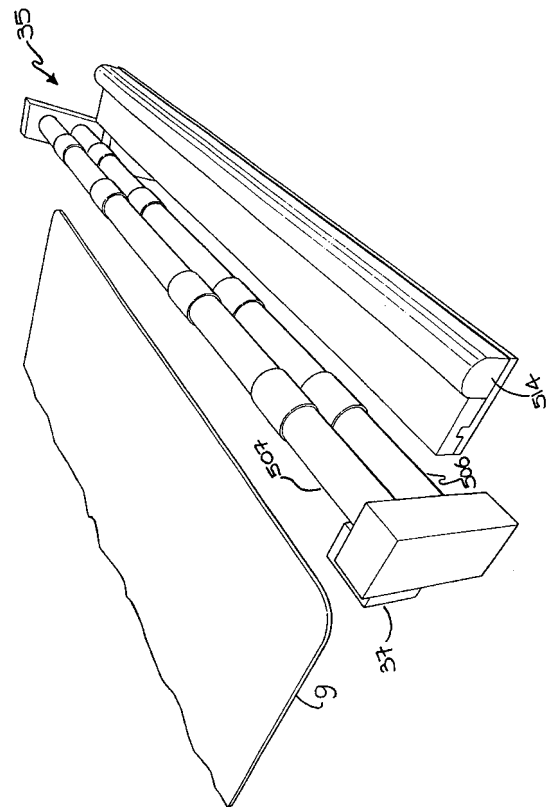
【図156】



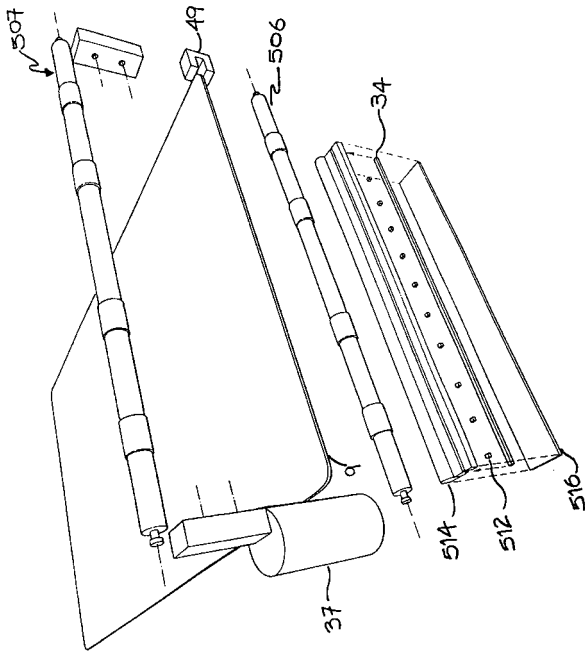
【図157】



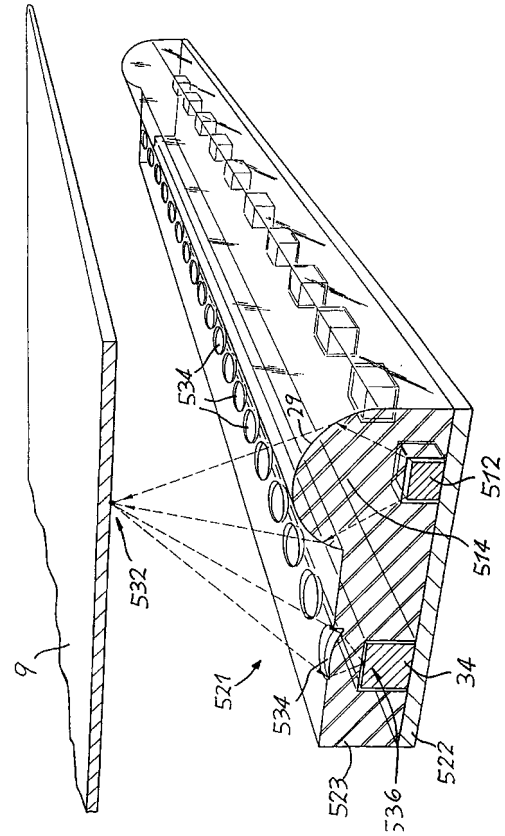
【図159】



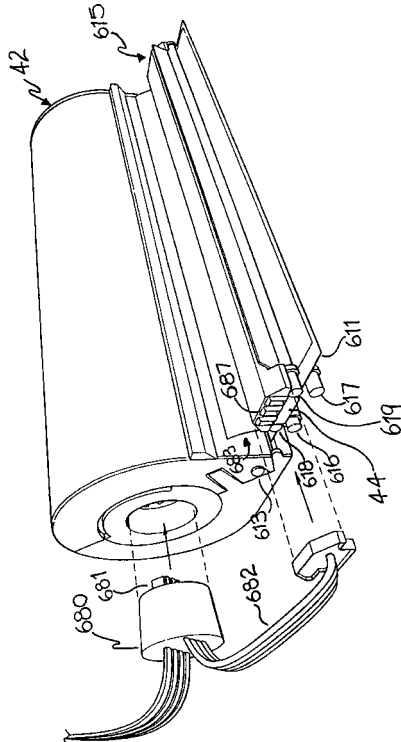
【図160】



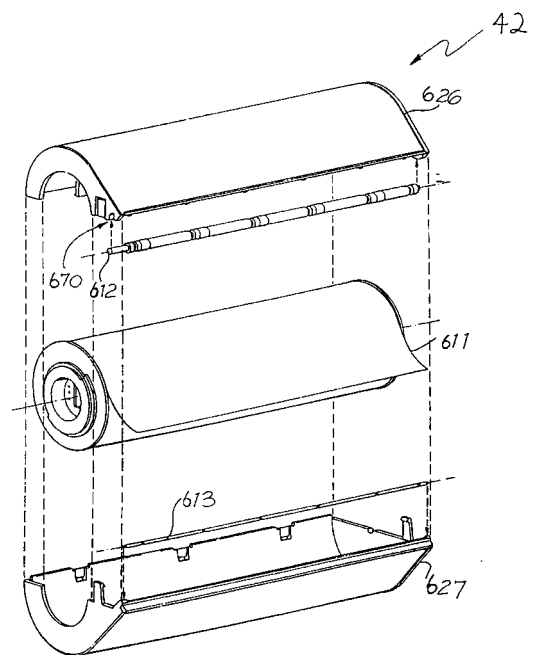
【図161】



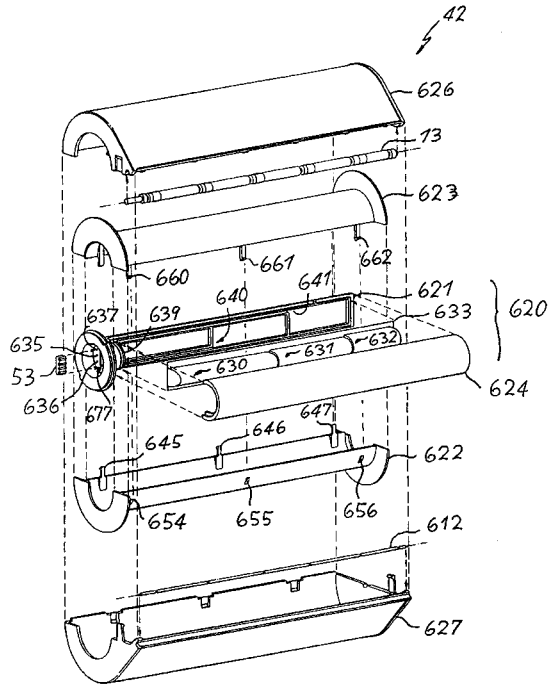
【図162】



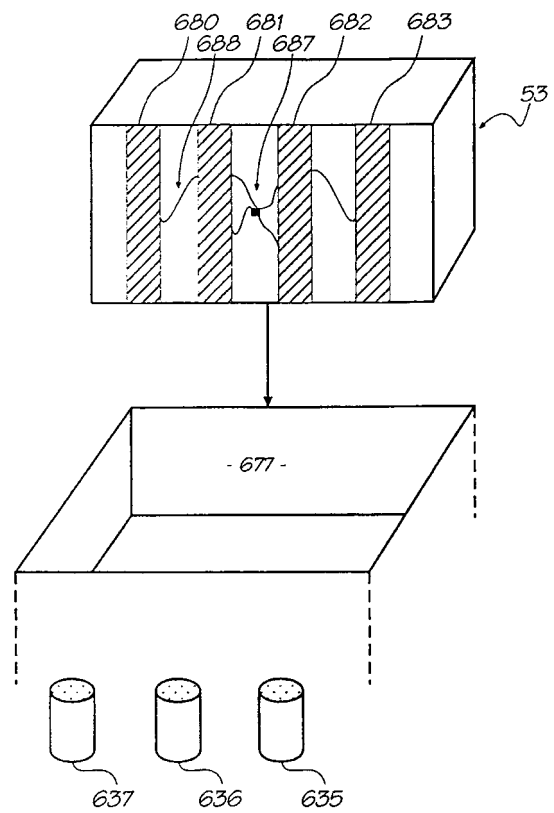
【図163】



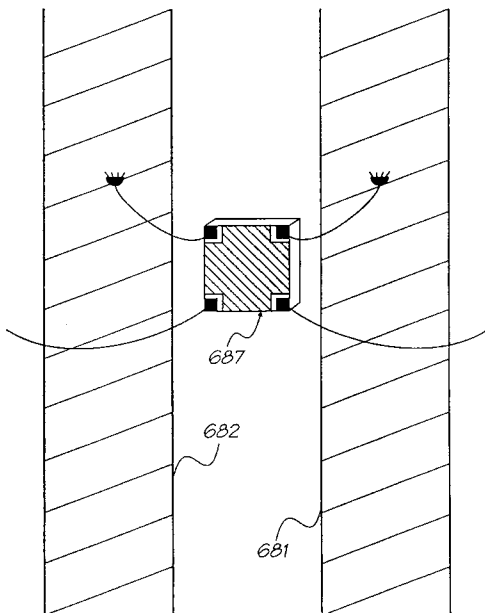
【図164】



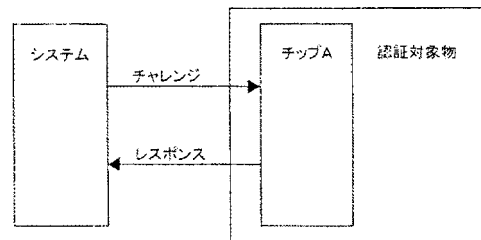
【図165】



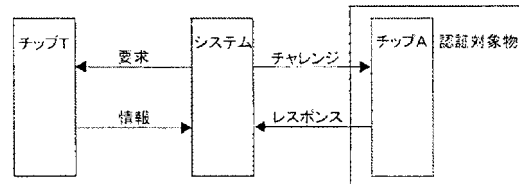
【図166】



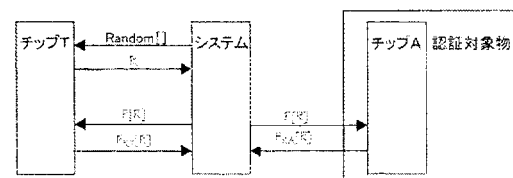
【図167】



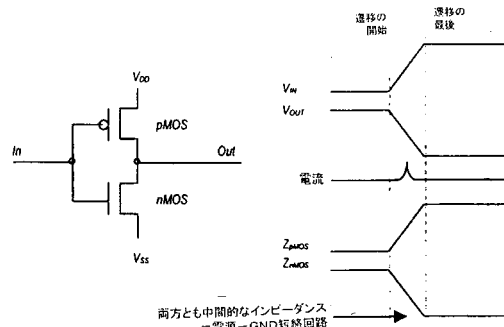
【図168】



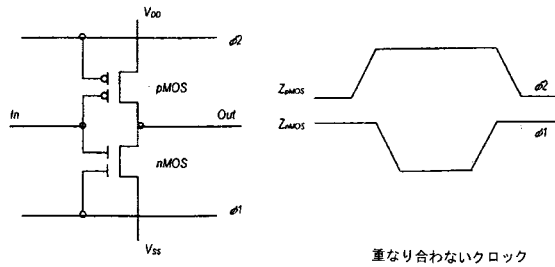
【図169】



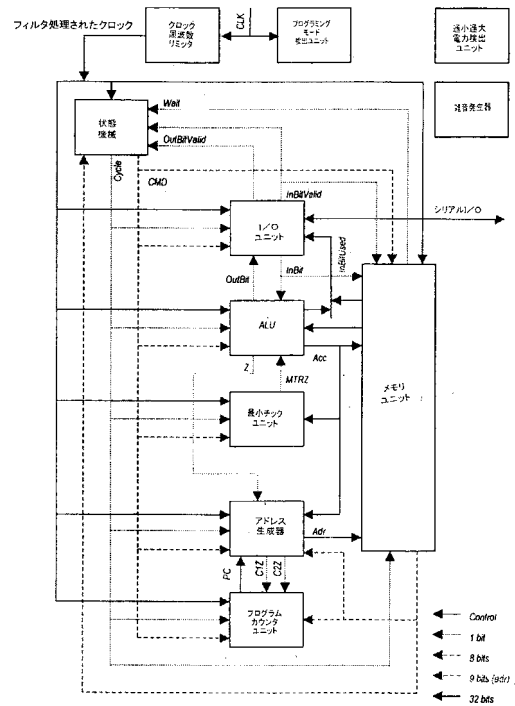
【図 179】



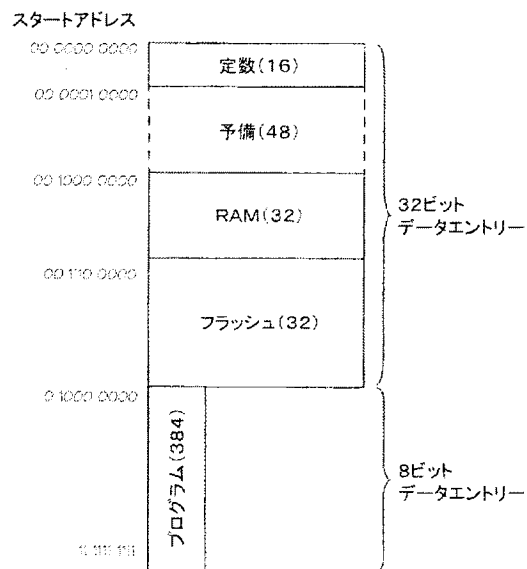
【図 180】



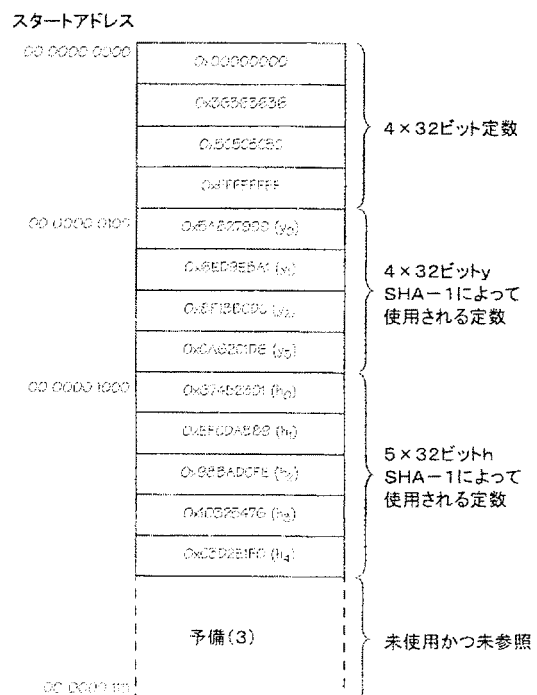
【図 181】



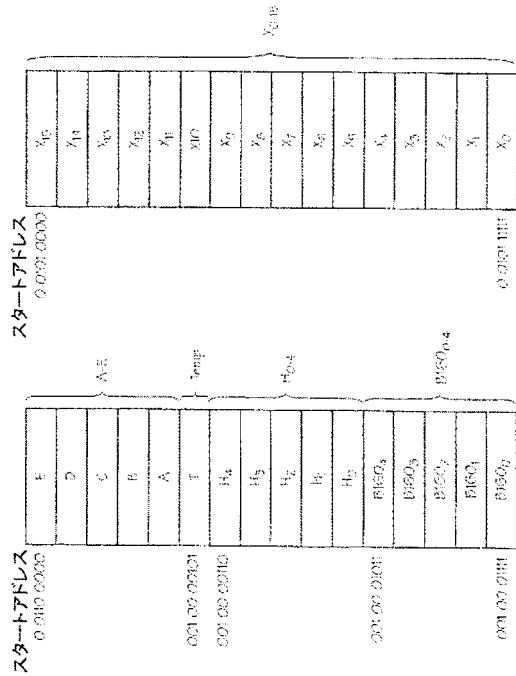
【図 182】



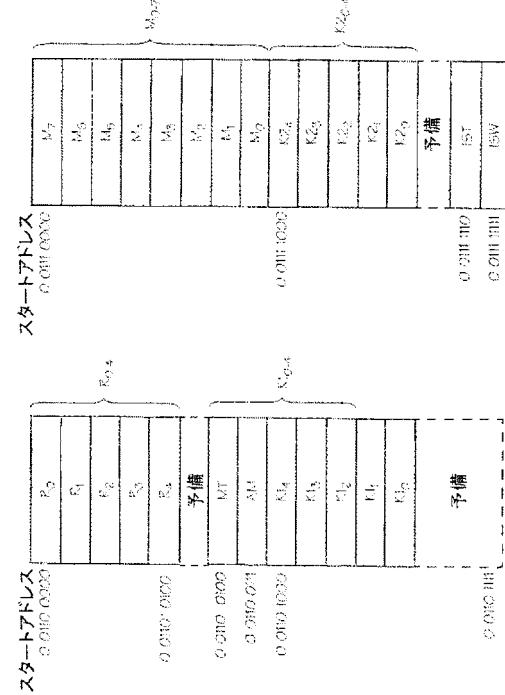
【図 183】



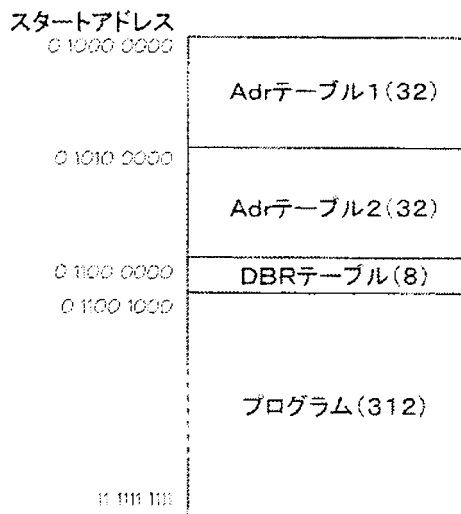
【図184】



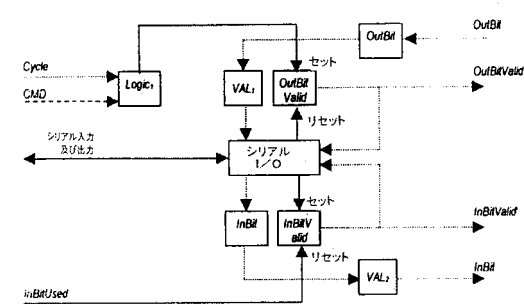
【図185】



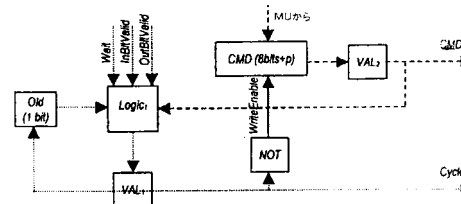
【図186】



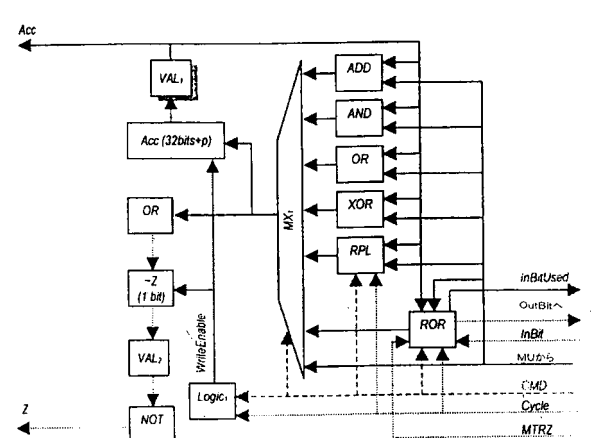
【図188】



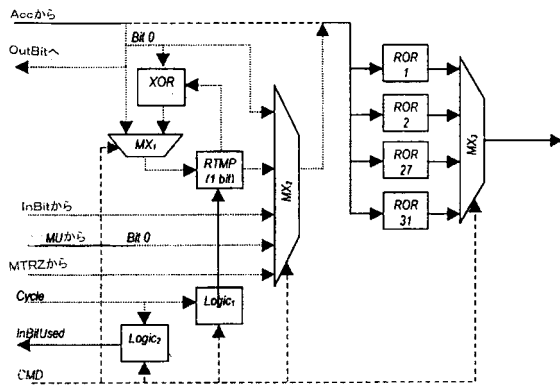
【図187】



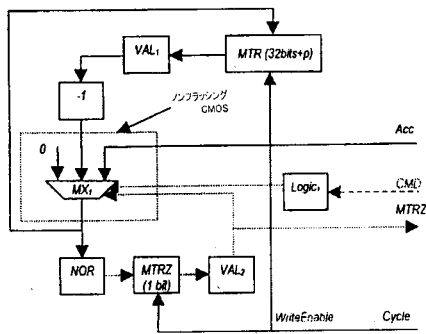
【図189】



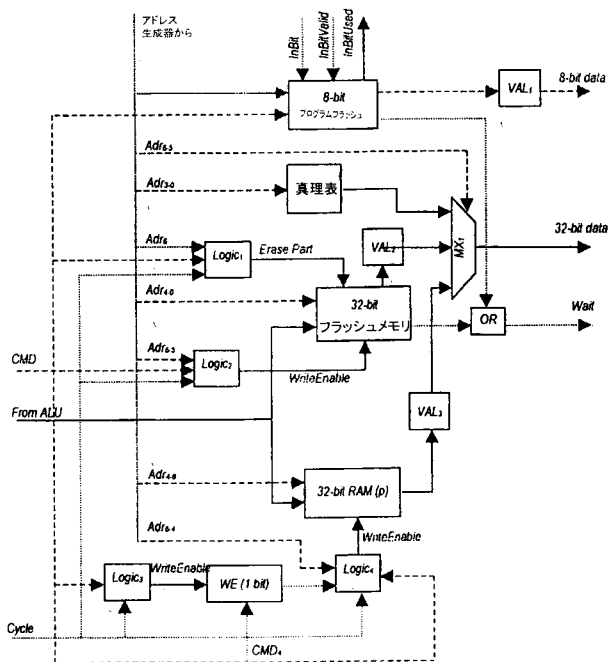
【图 192】



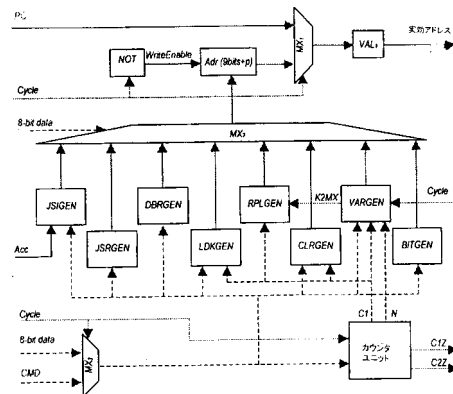
【 図 1 9 1 】



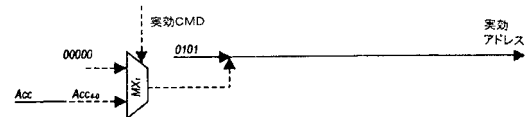
【図 193】



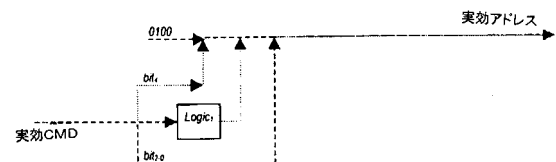
【图 194】



【図 1 9 5】



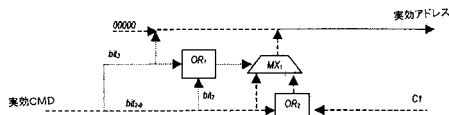
【図 196】



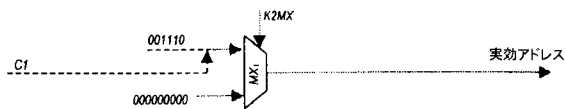
【図 197】



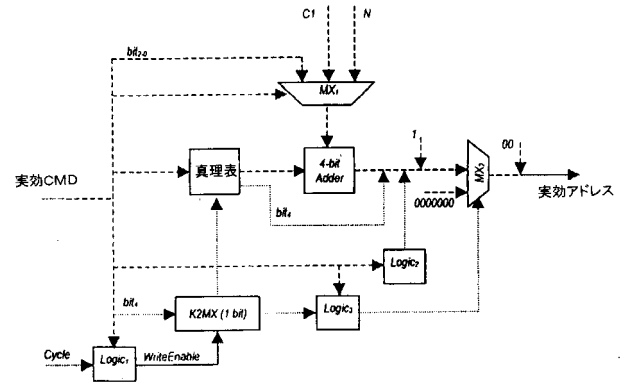
【図 198】



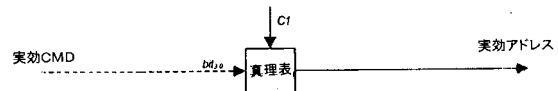
【図 199】



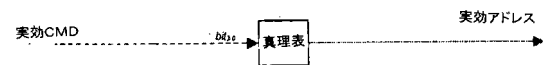
【図 200】



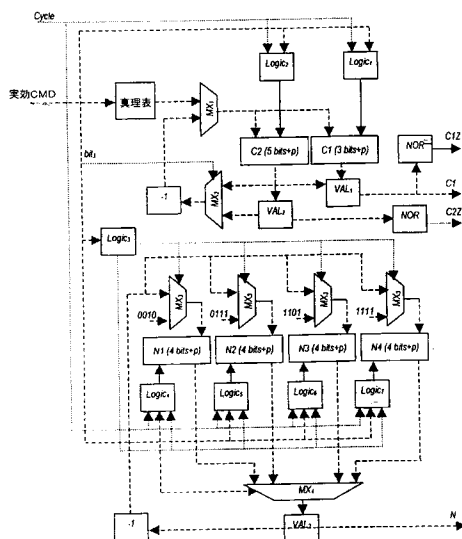
【図 201】



【図 202】



【図 203】



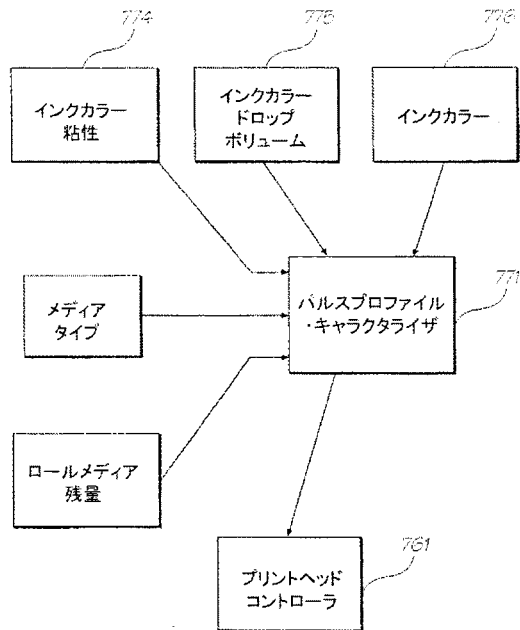
【図 204】

705

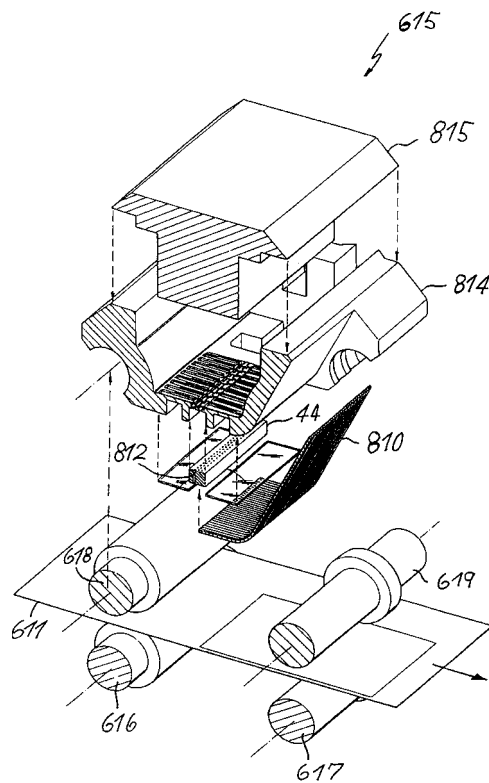
データタイプ	ビット数
工場コード	16
パッチ番号	32
シリアル番号	48
製造日	16
メディア長さ	24
メディアタイプ	8
プレ印刷メディア長さ	16
シアンインク粘性	8
マゼンタインク粘性	8
イエローインク粘性	8
シアンドロップボリューム	8
マゼンタドロップボリューム	8
イエロードロップボリューム	8
シアンインクカラー	24
マゼンタインクカラー	24
イエローインクカラー	24
残りメディア長さインジケータ	16
認証鍵	128
著作権保護可能ビットパターン	512
カメラ用予備	88
合計	1024

728

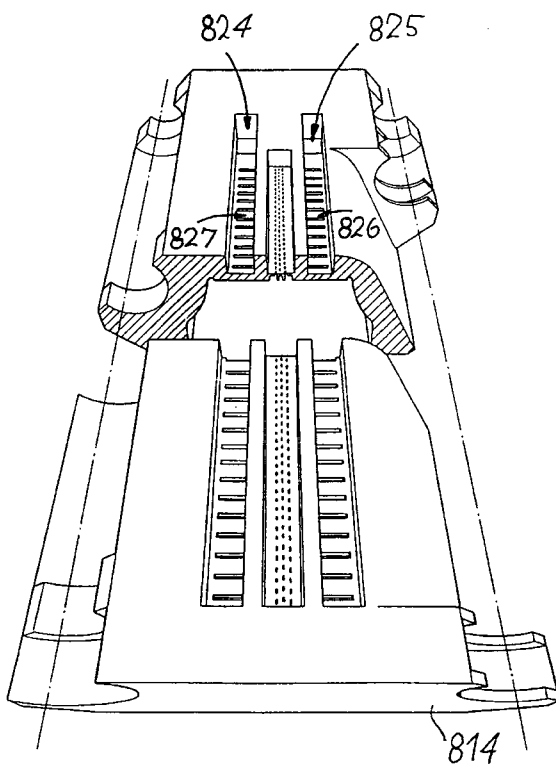
【図 205】



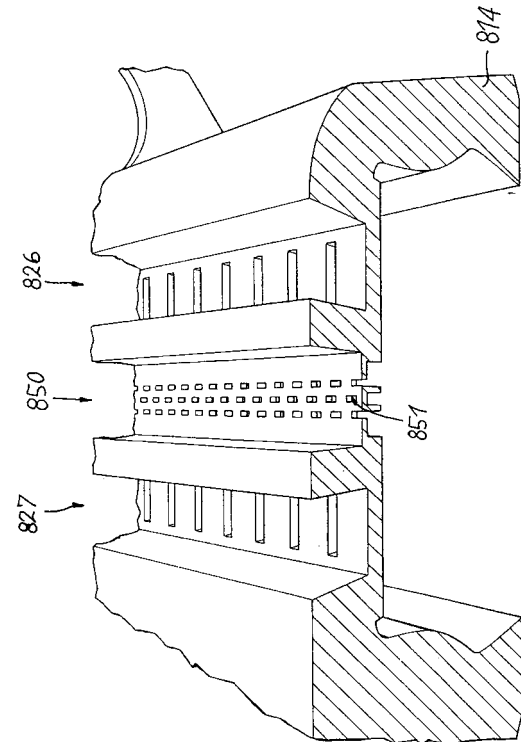
【図 206】



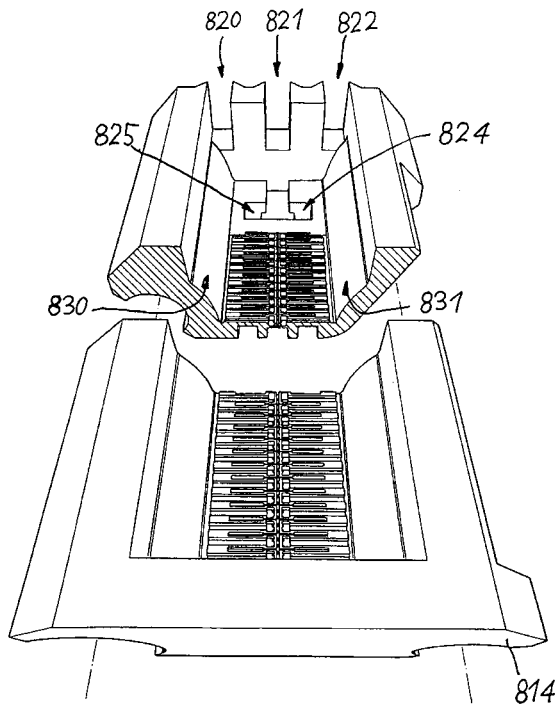
【図 207】



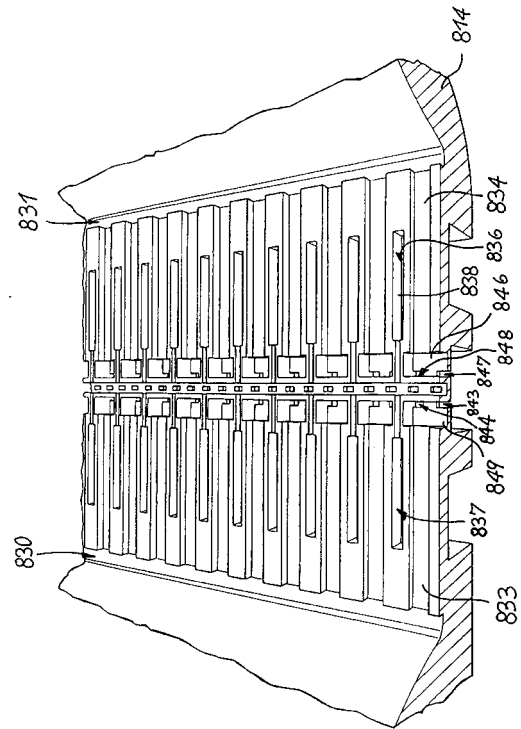
【図 208】



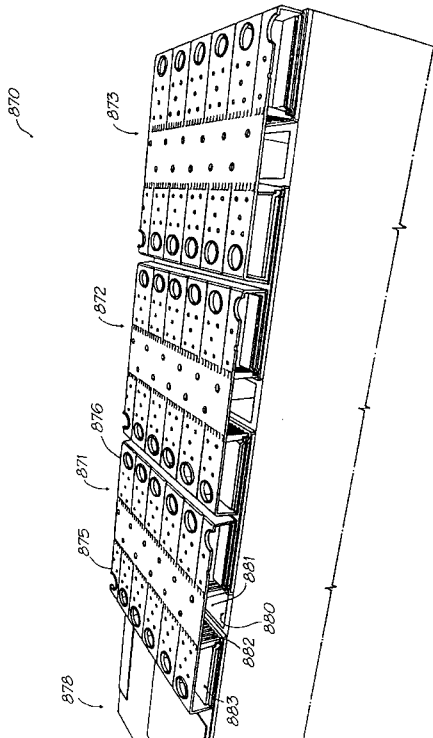
【図 209】



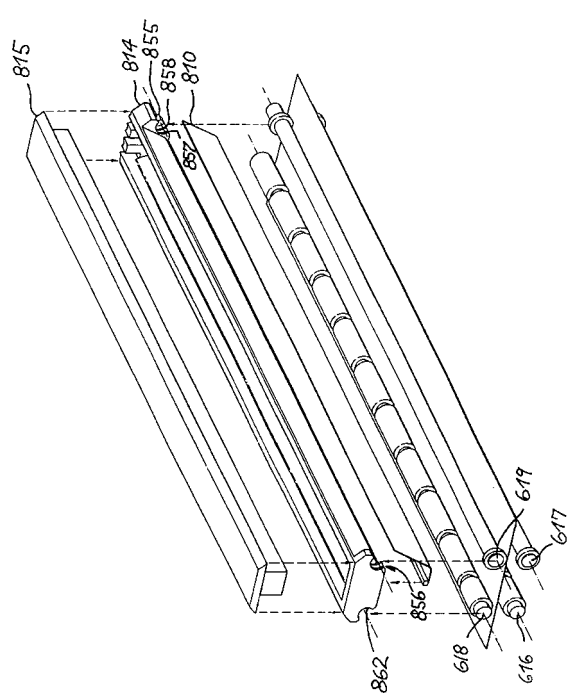
【図 210】



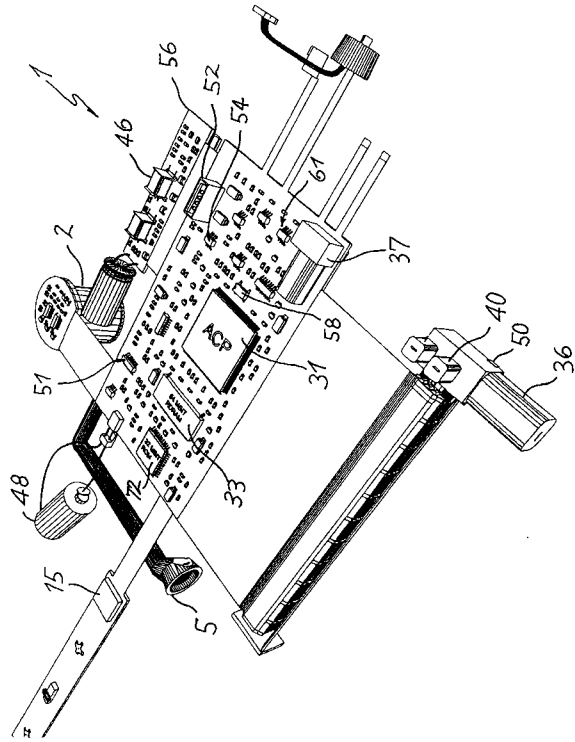
【図 211】



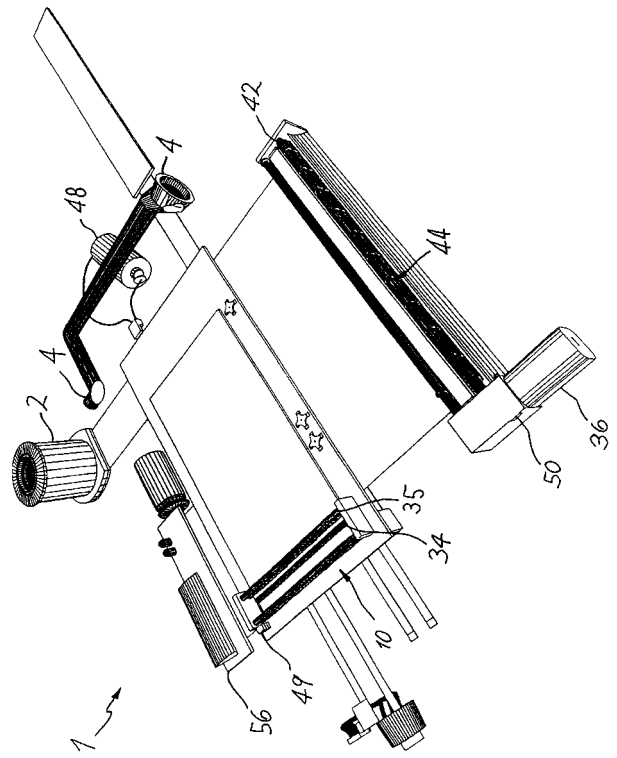
【図 212】



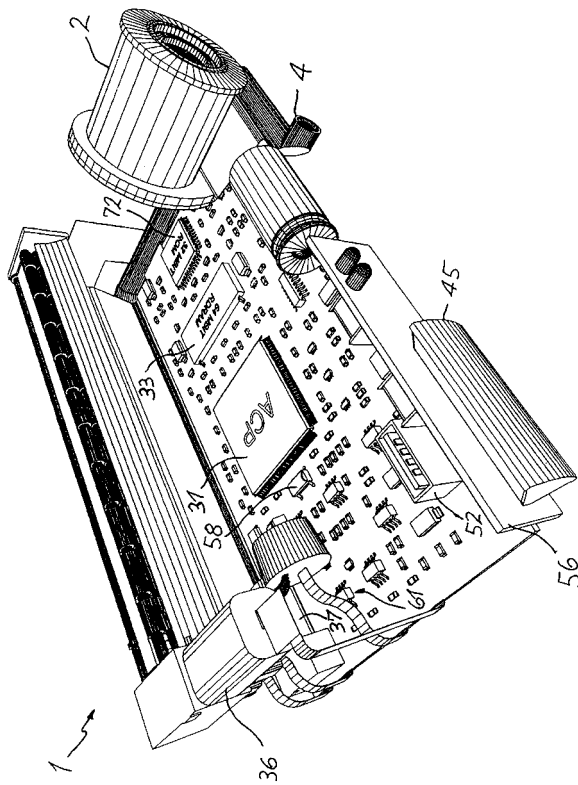
【図 2 1 3】



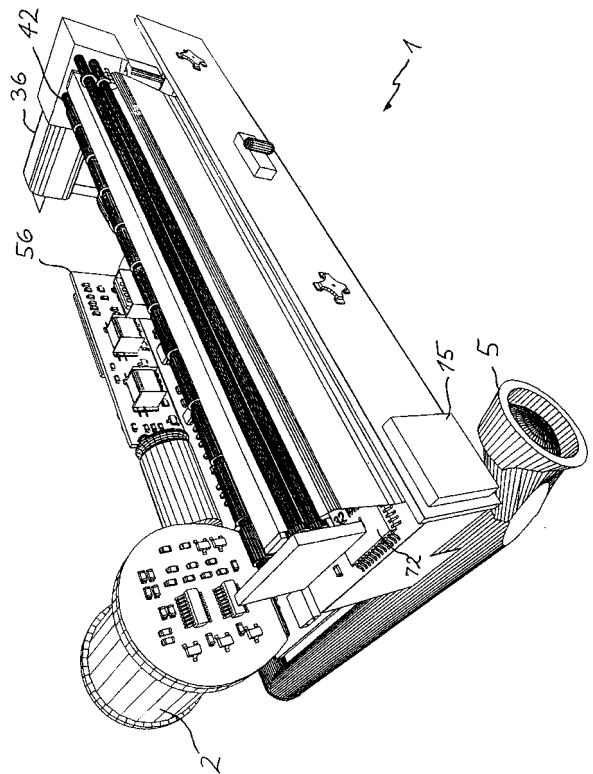
【図 2 1 4】



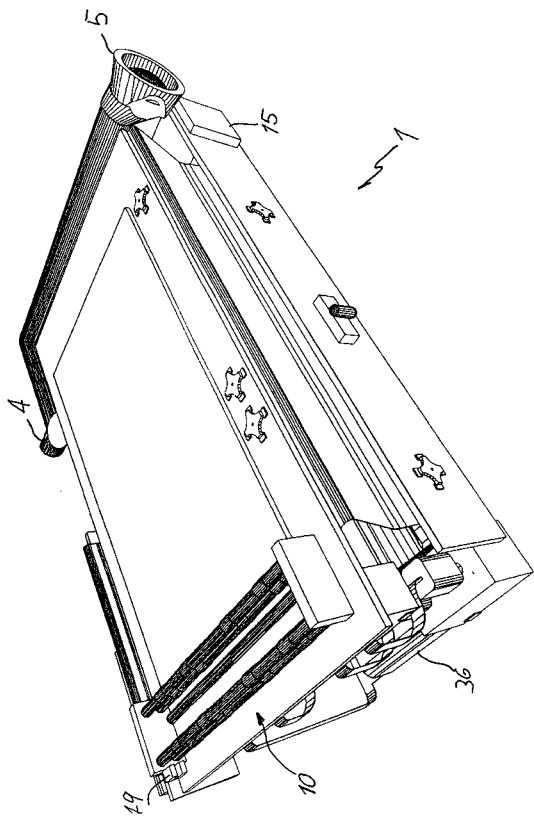
【図 2 1 5】



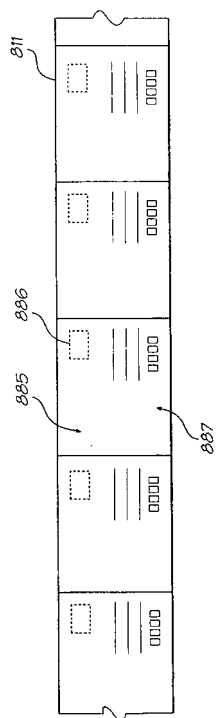
【図 2 1 6】



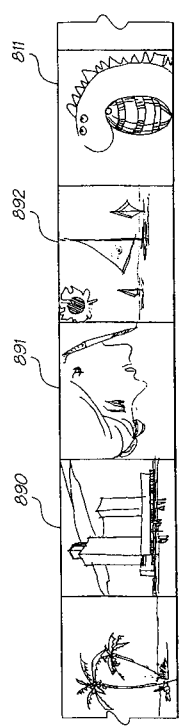
【図 2 1 7】



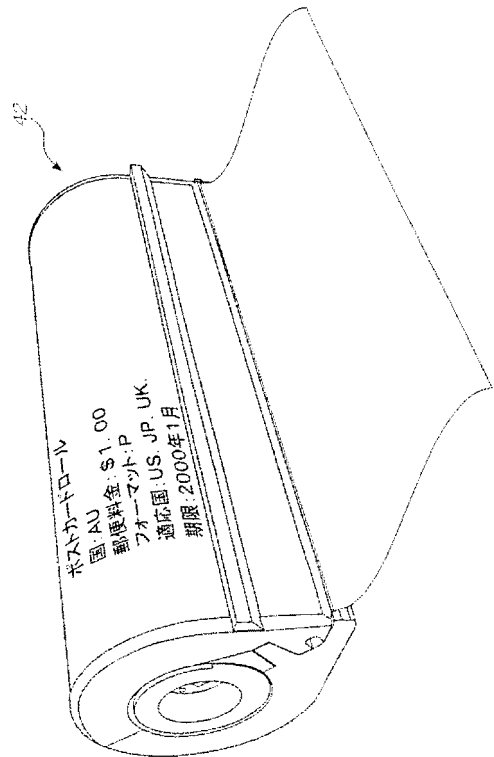
【図 2 1 8】



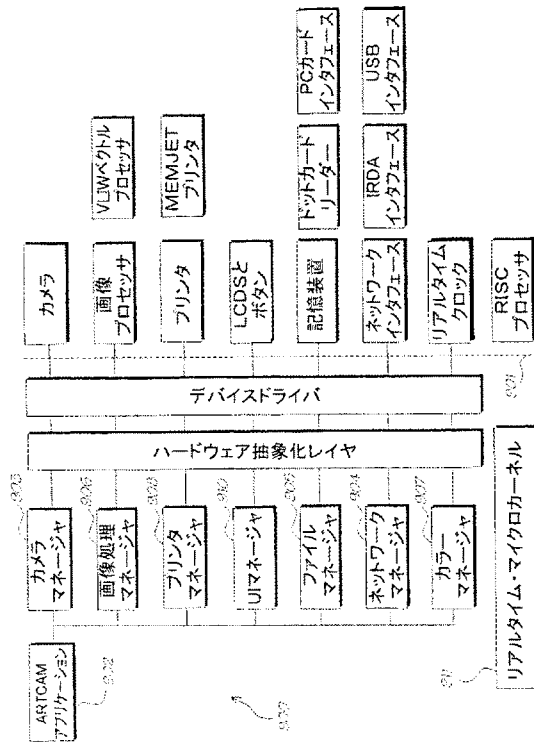
【図 2 1 9】



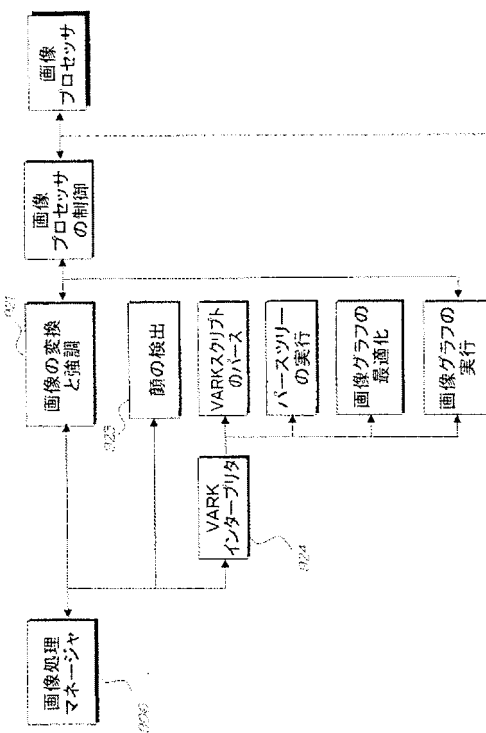
【図 2 2 0】



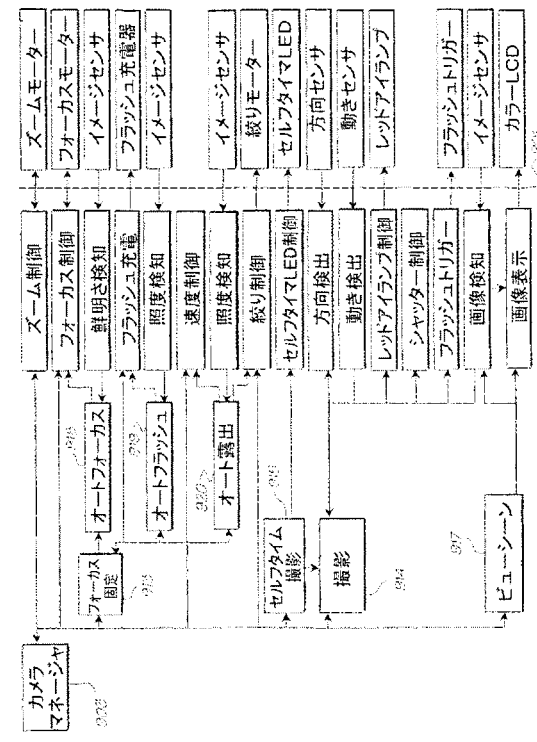
【図 2 2 1】



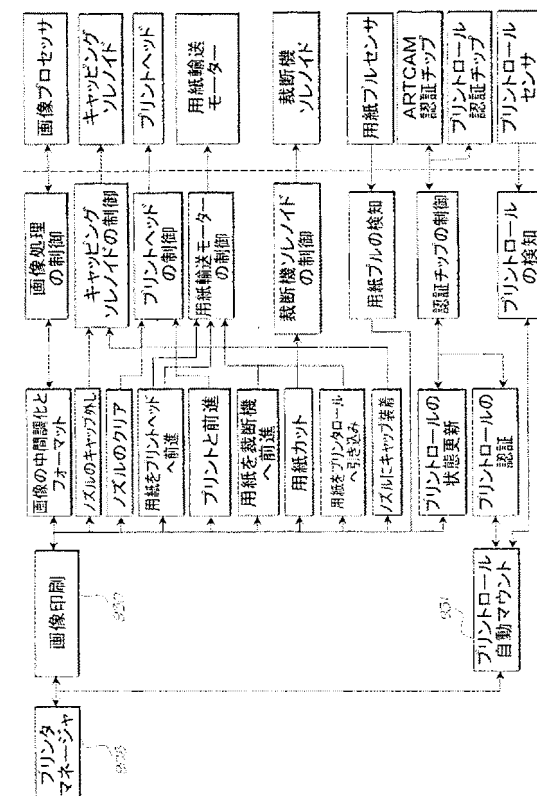
【図 2 2 3】



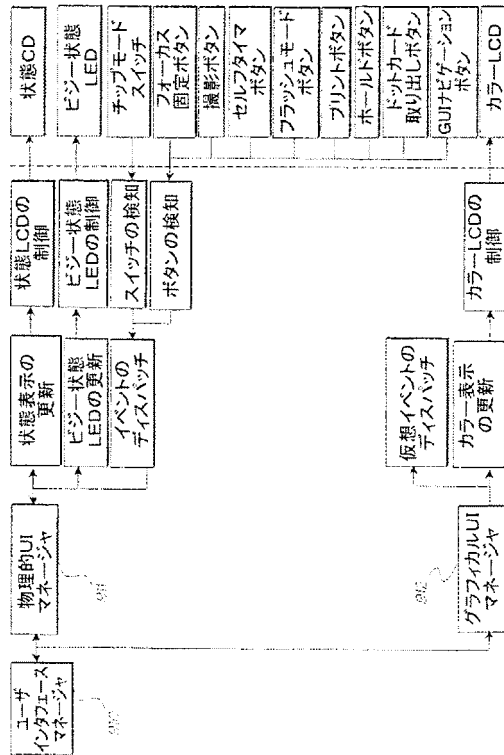
【図 2 2 2】



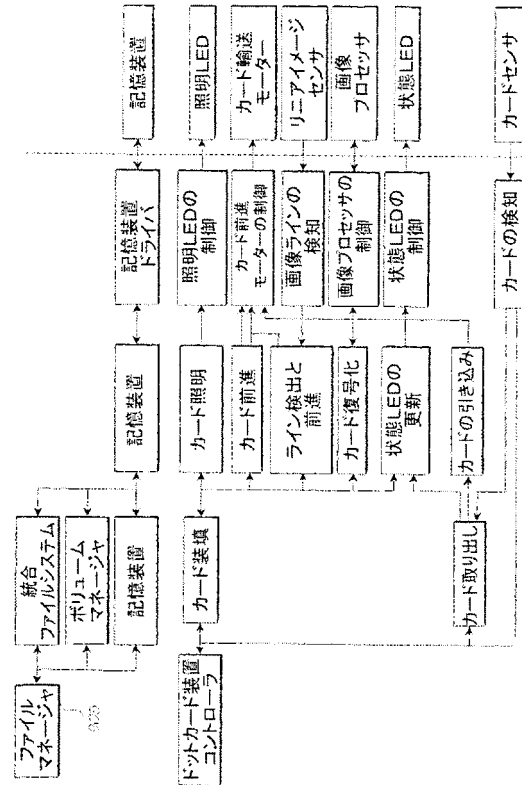
【図 2 2 4】



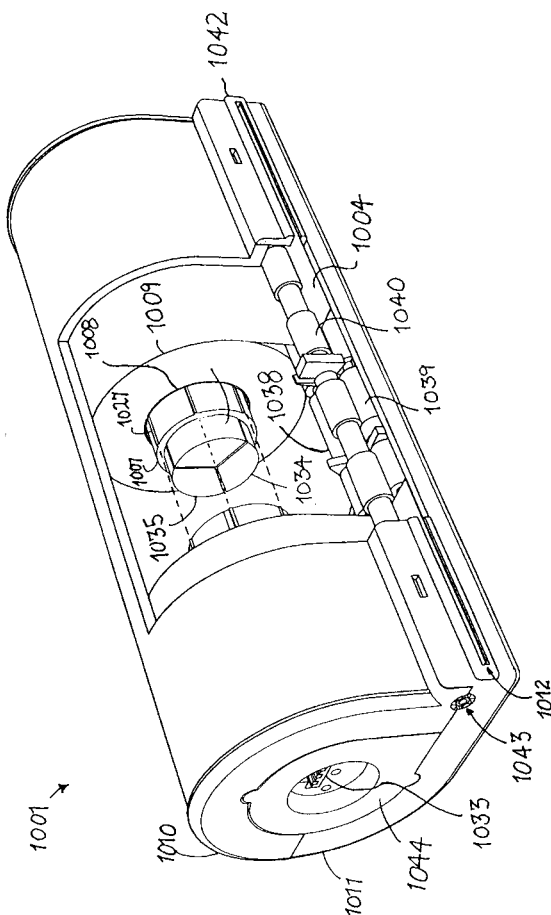
【図 2 2 5】



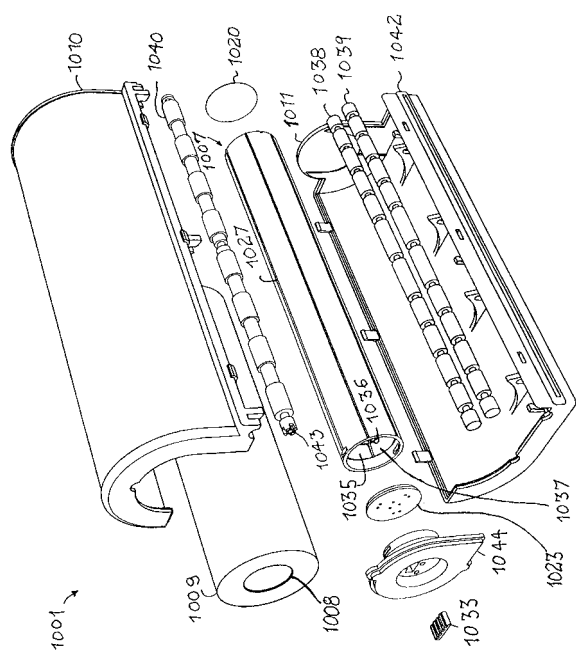
【図 2 2 6】



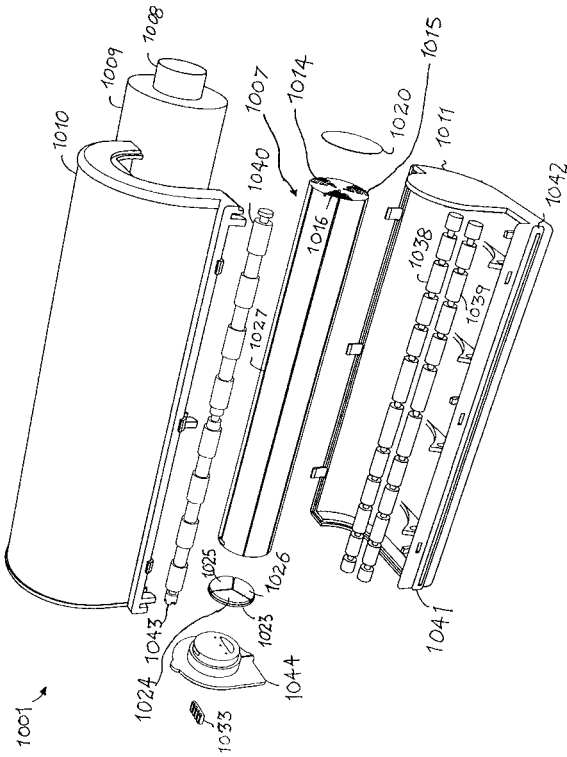
【図 2 2 7】



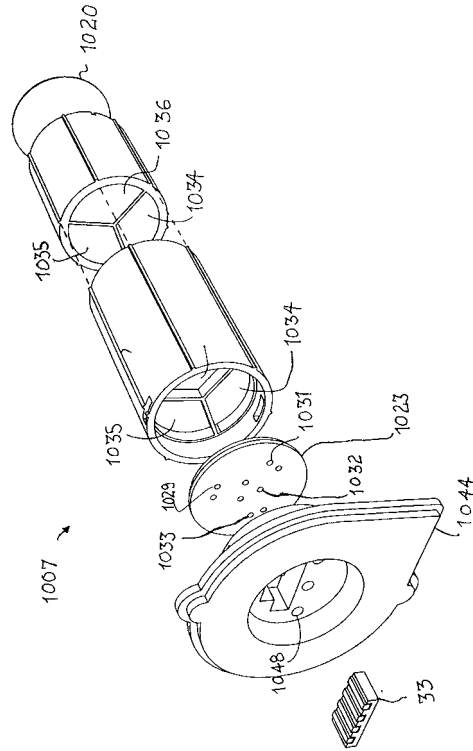
【図 2 2 8】



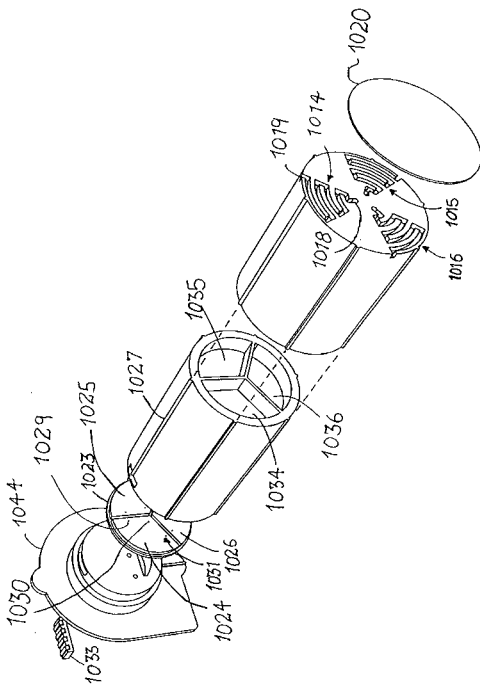
【図 2 2 9】



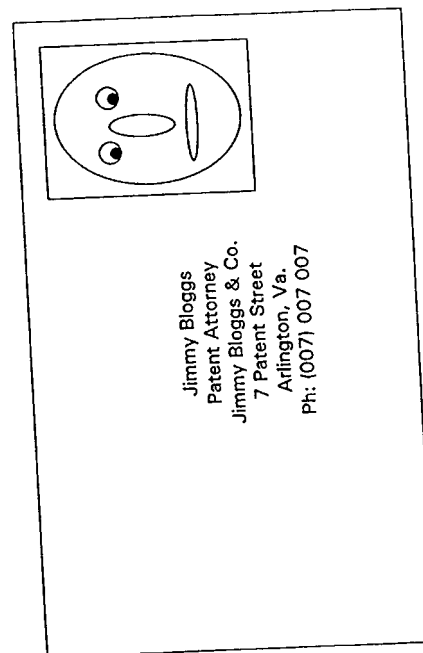
【図 2 3 0】



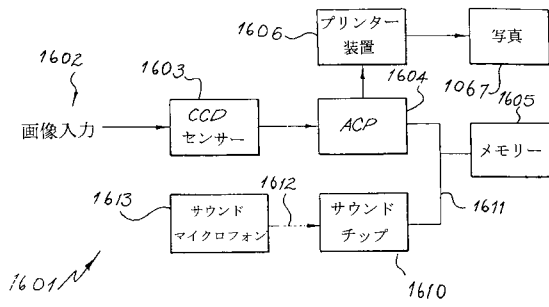
【図 2 3 1】



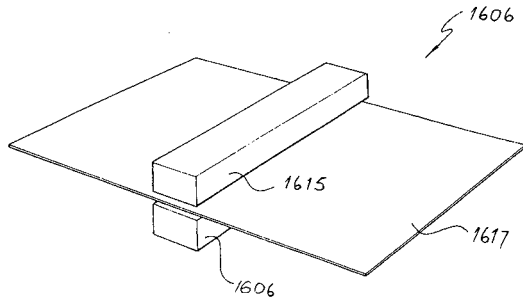
【図 2 3 2】



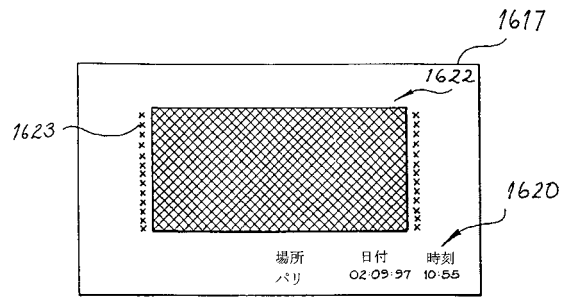
【図 2 3 3】



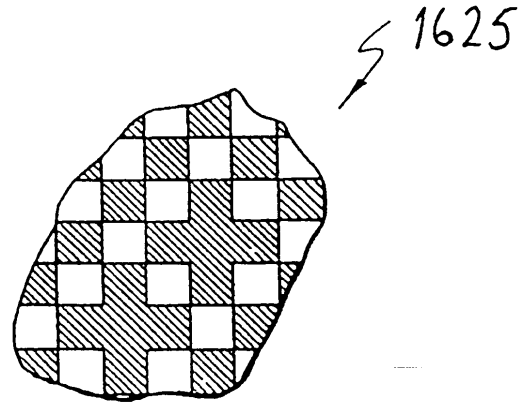
【図 2 3 4】



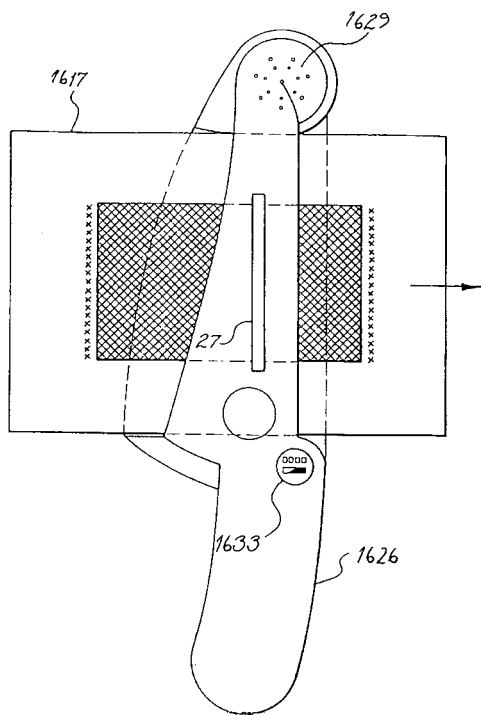
【図 2 3 5】



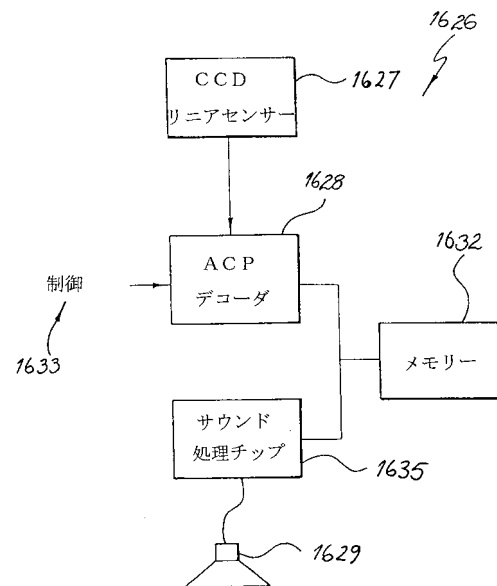
【図 2 3 6】



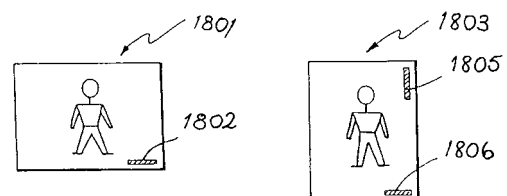
【図 2 3 7】



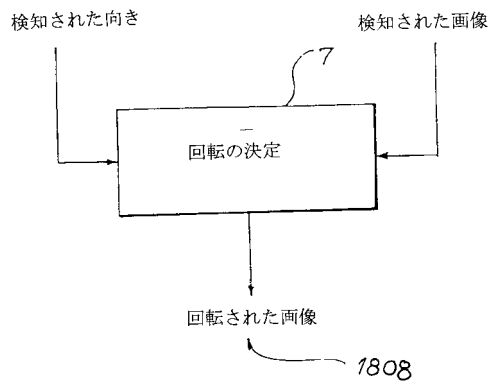
【図 2 3 8】



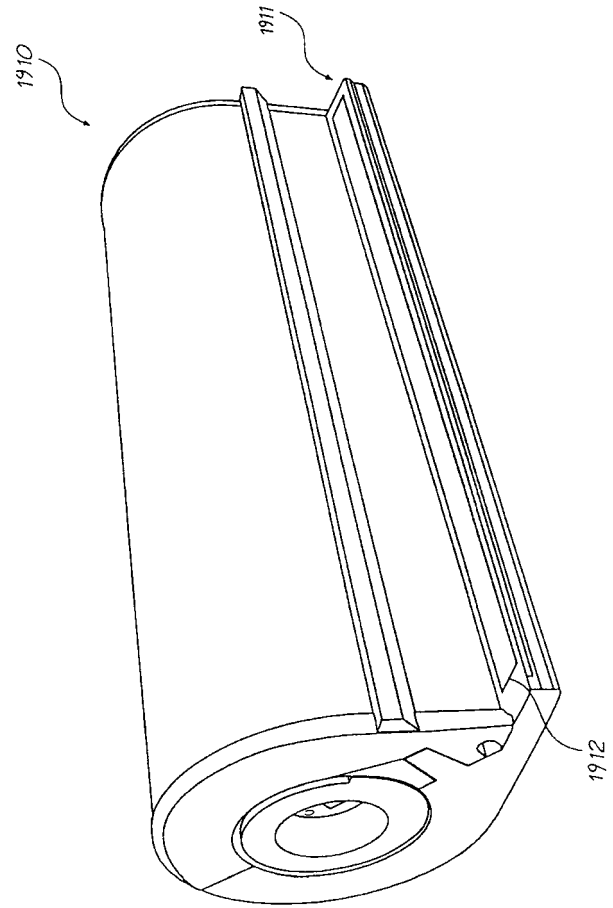
【図 2 3 9】



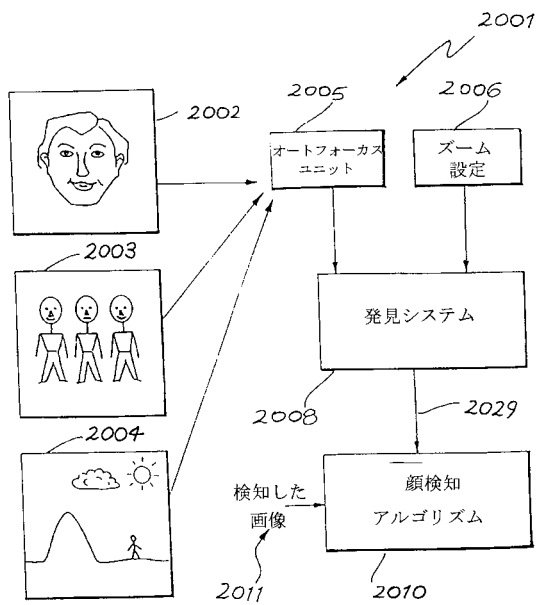
【図 2 4 0】



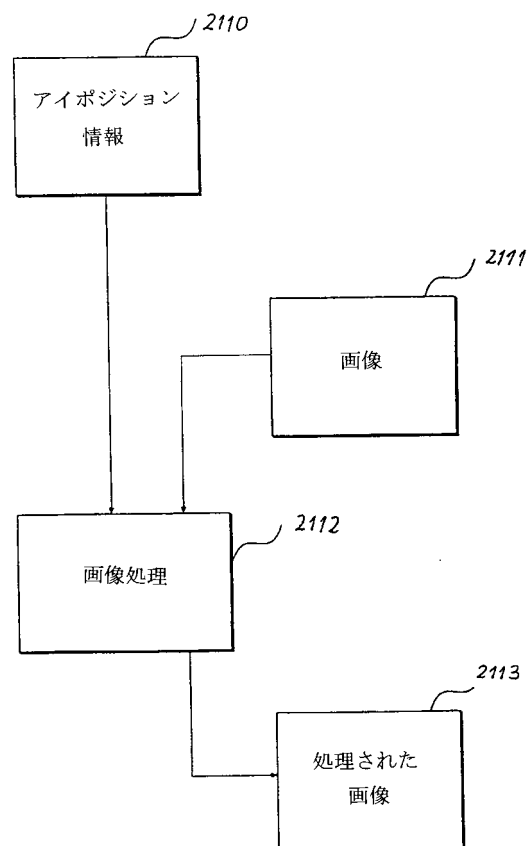
【図 2 4 1】



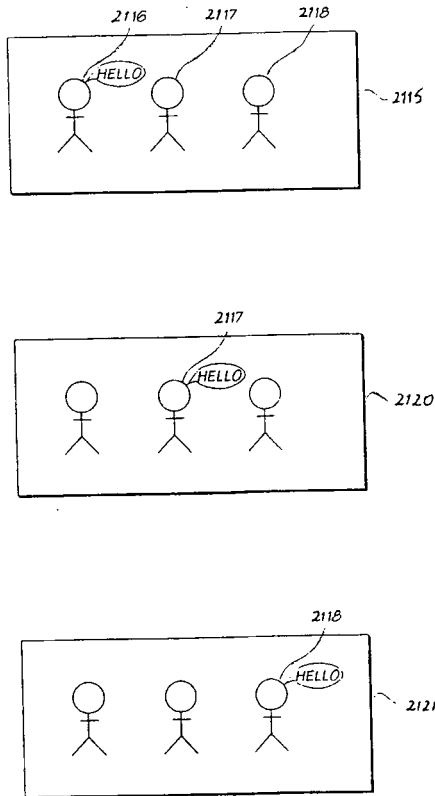
【図 2 4 2】



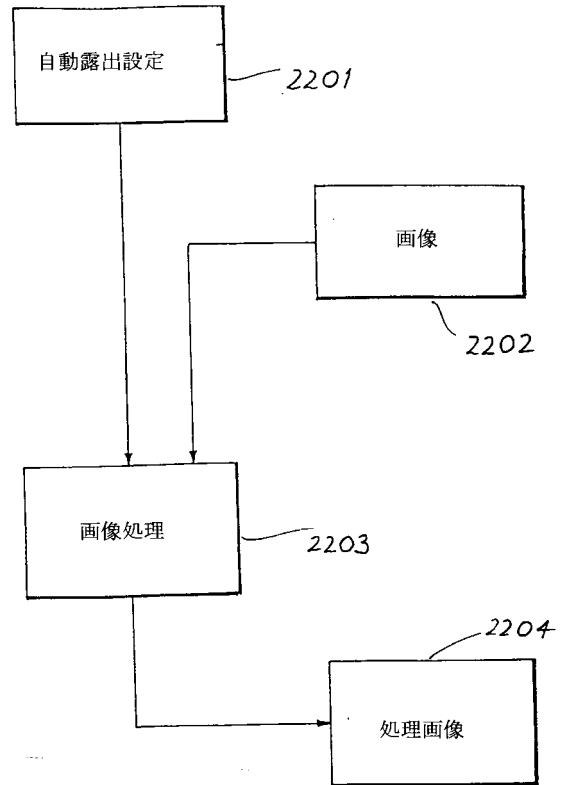
【図 2 4 3】



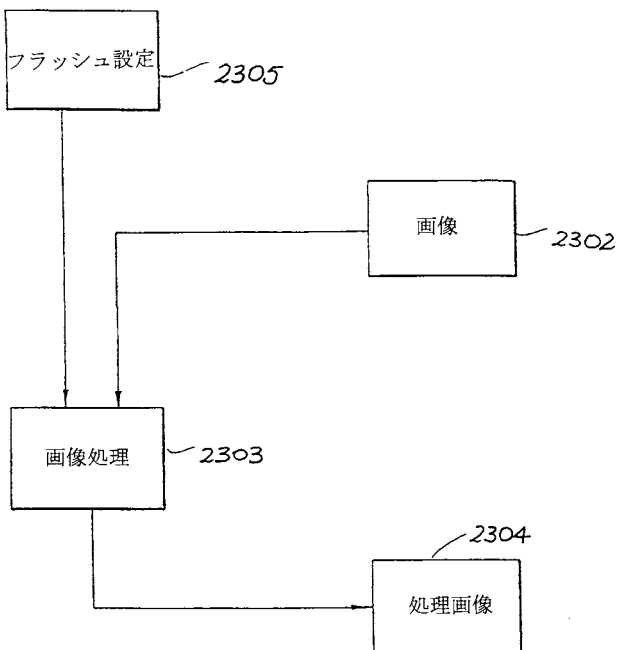
【図 2 4 4】



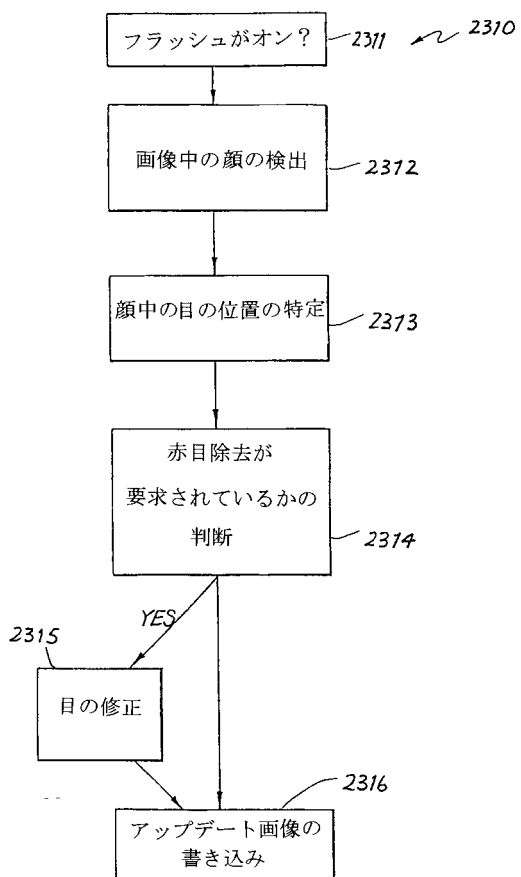
【図 2 4 5】



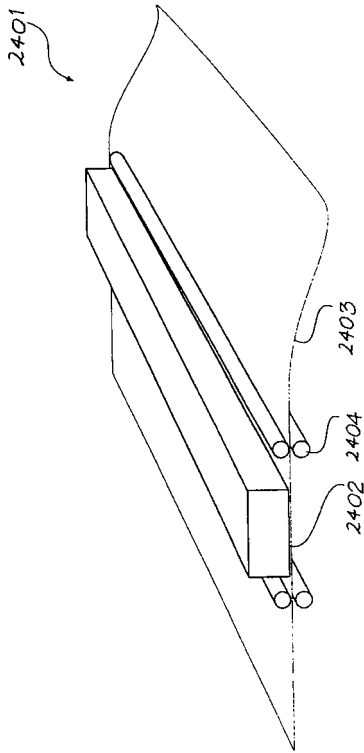
【図 2 4 6】



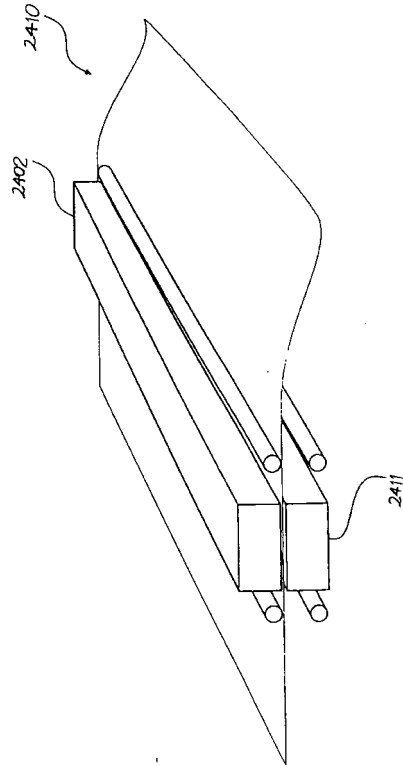
【図 2 4 7】



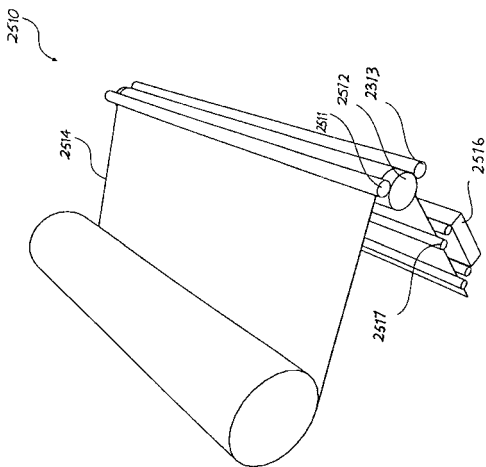
【図 248】



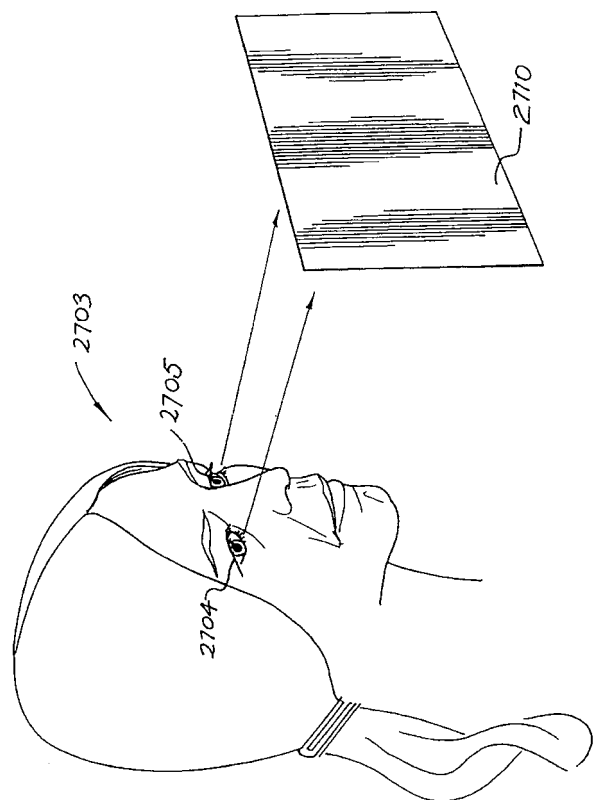
【図 249】



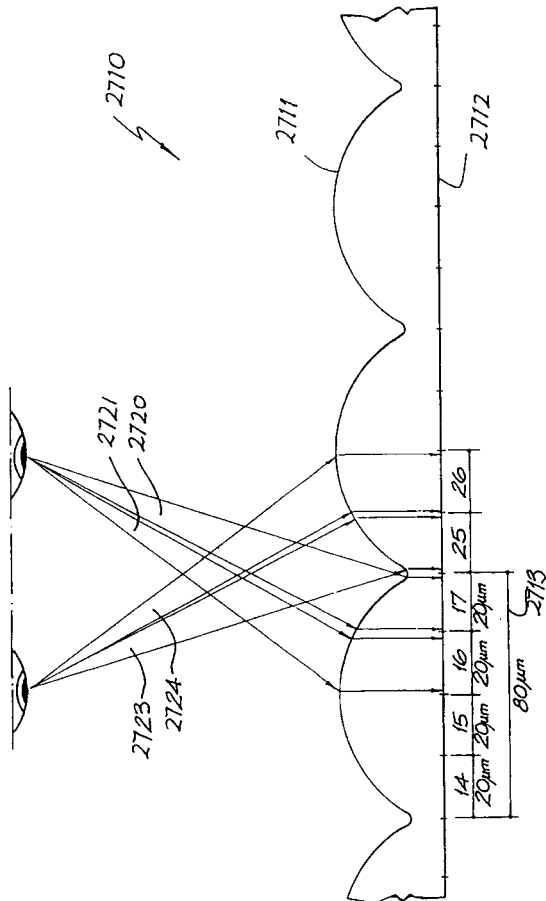
【図 250】



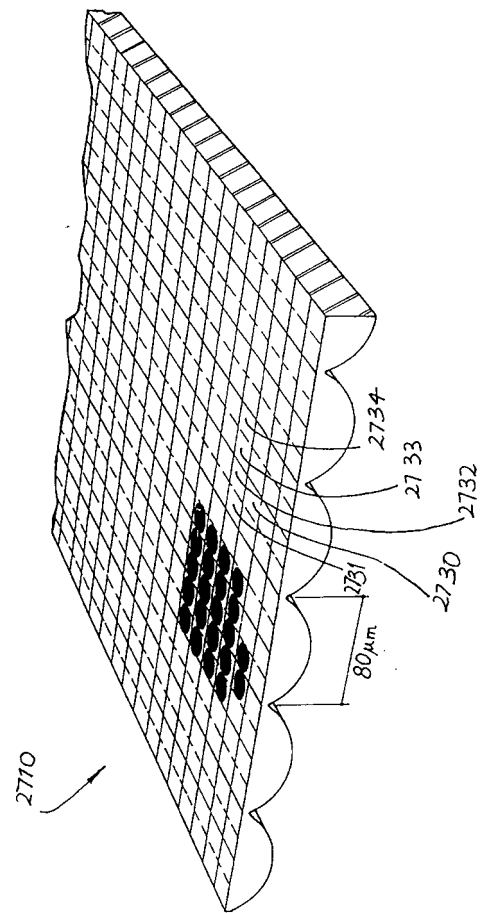
【図 251】



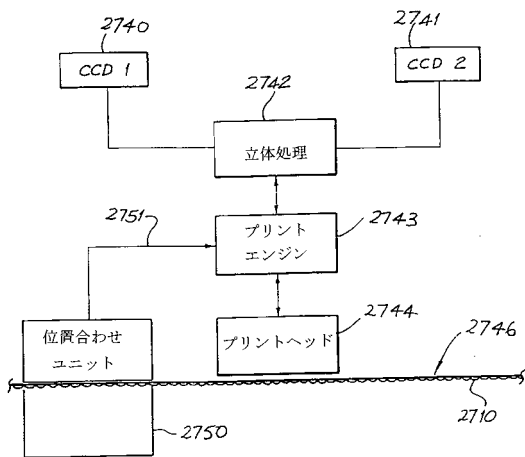
【 図 2 5 2 】



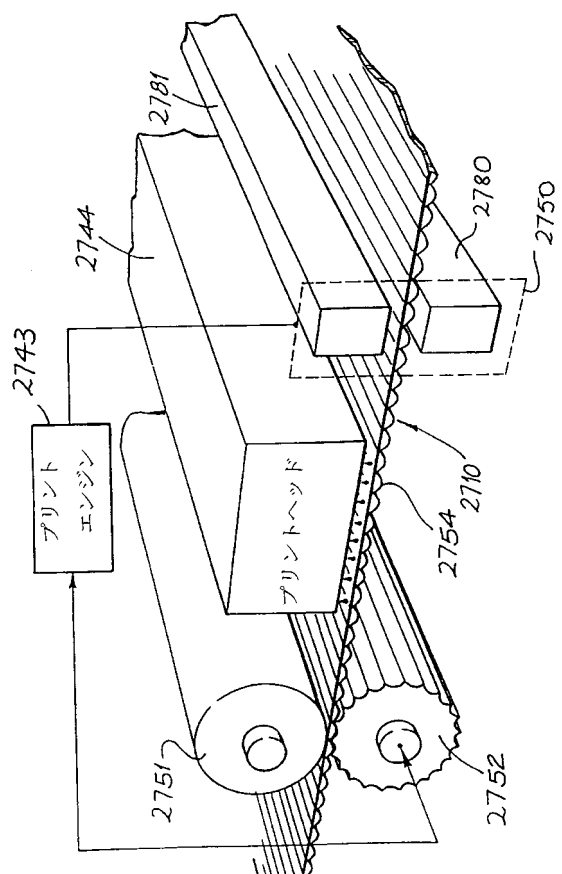
【图 2 5 3】



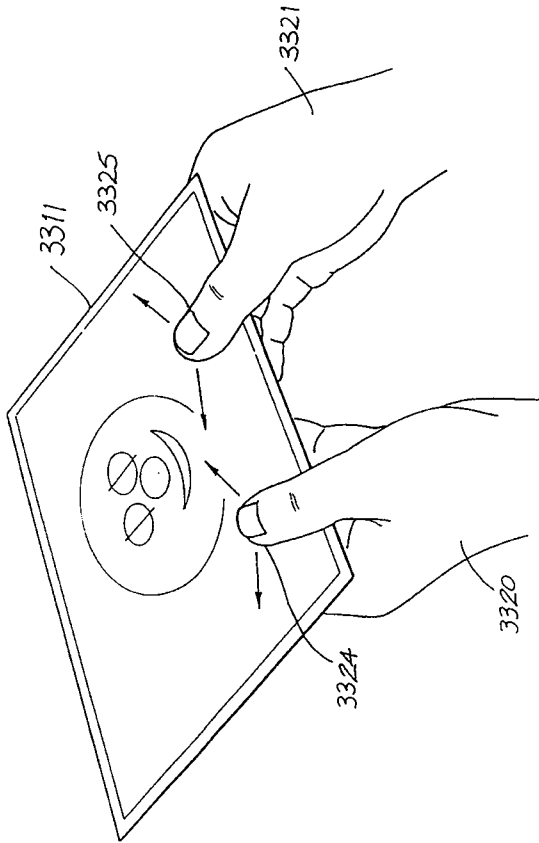
【 図 2 5 4 】



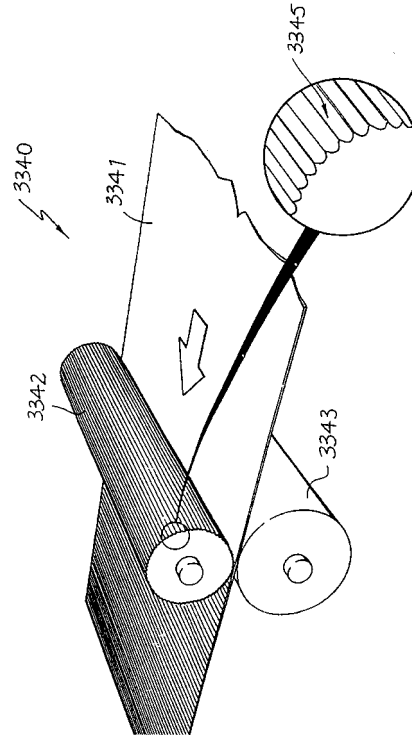
【図 2 5 5】



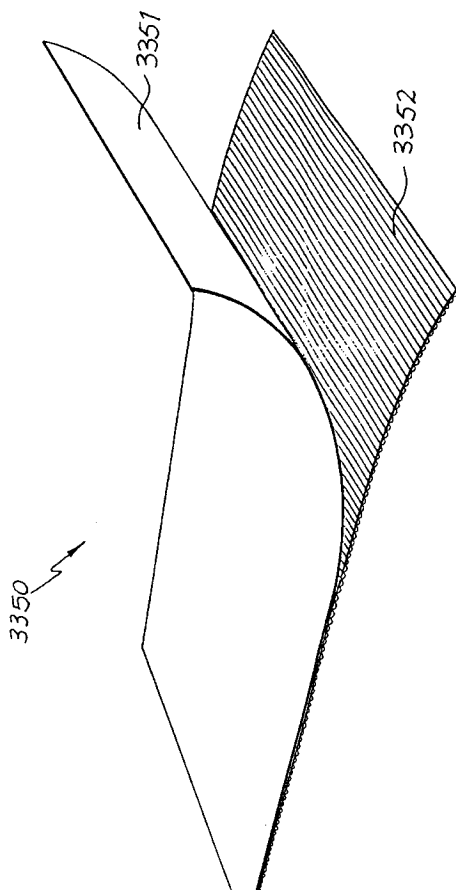
【図 260】



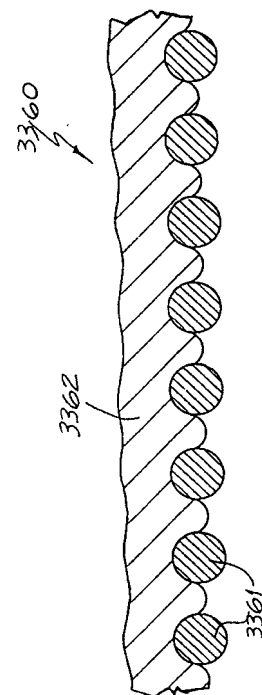
【図 261】



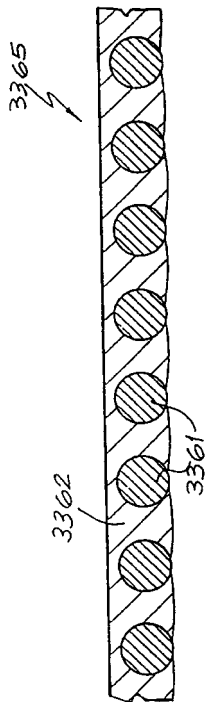
【図 262】



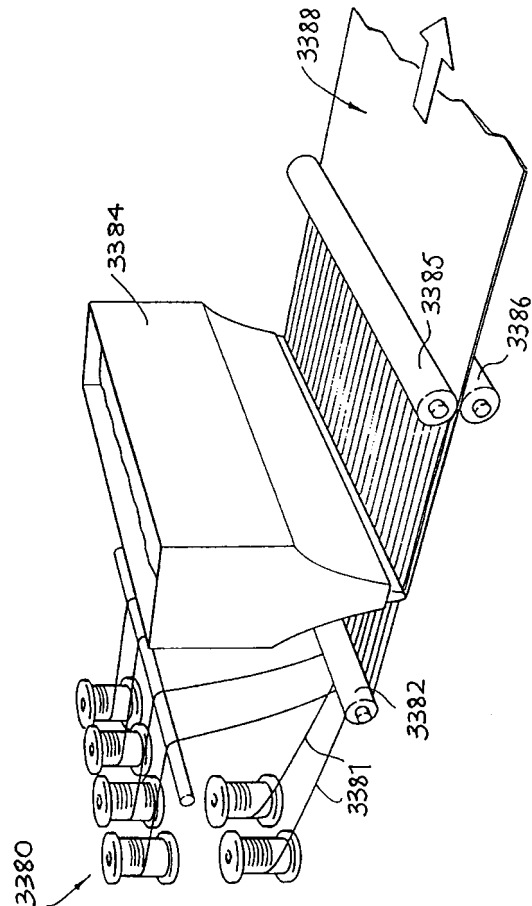
【図 263】



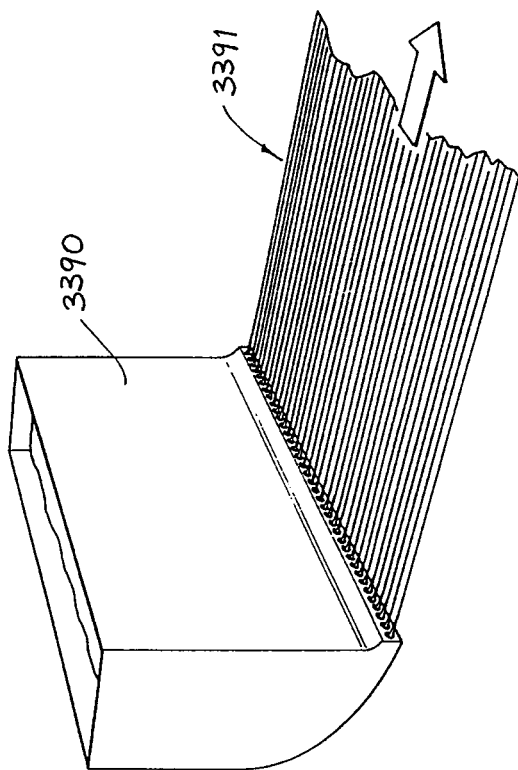
【図 2 6 4】



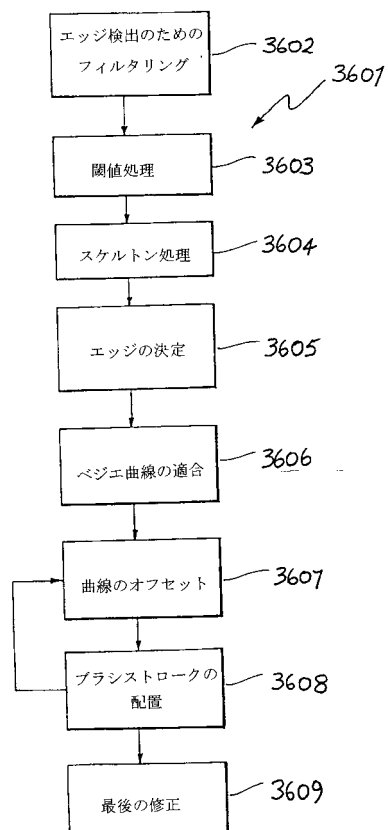
【図 2 6 5】



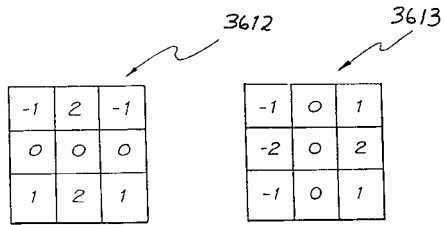
【図 2 6 6】



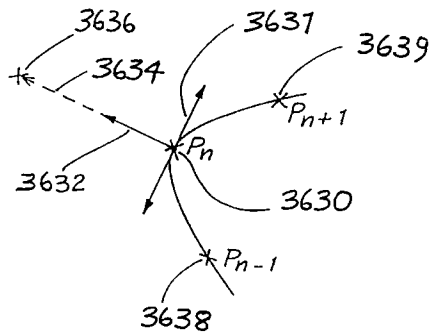
【図 2 6 7】



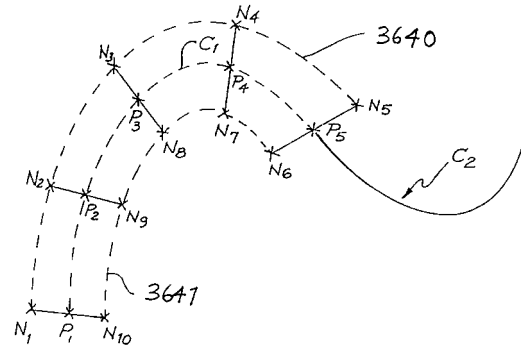
【図 2 6 8】



【図 2 6 9】



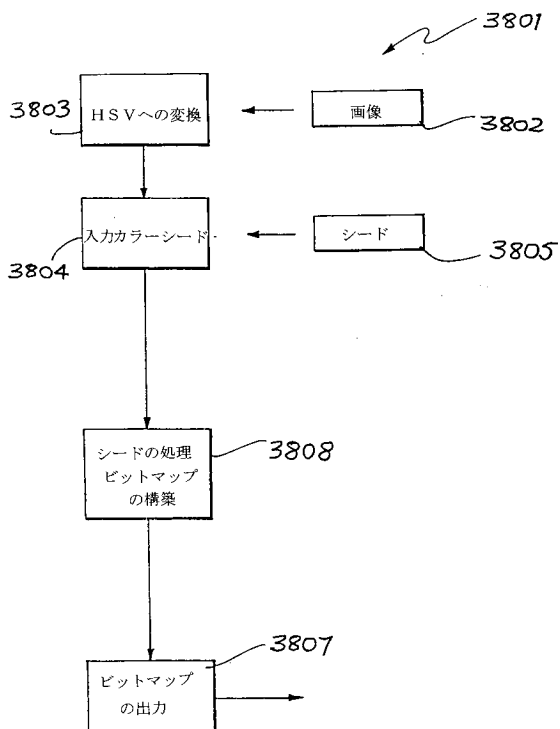
【図 2 7 0】



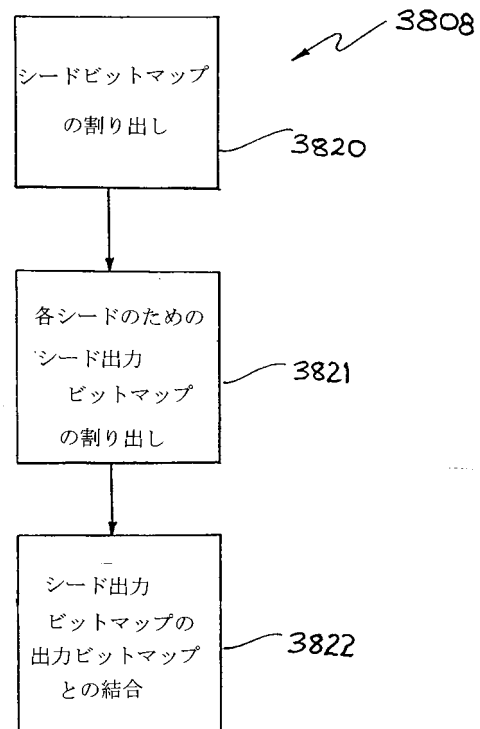
【図 2 7 1】



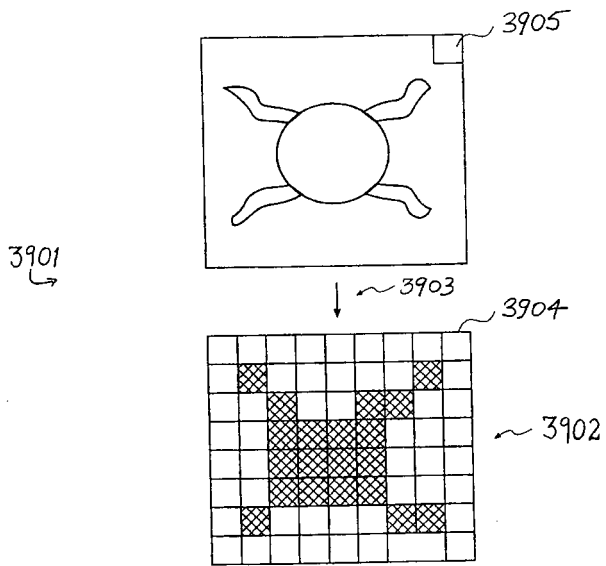
【図 2 7 2】



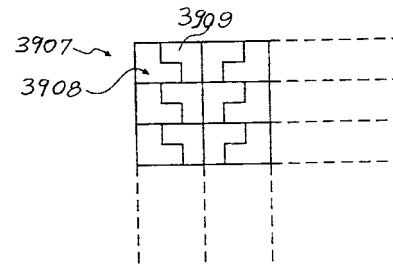
【図 2 7 3】



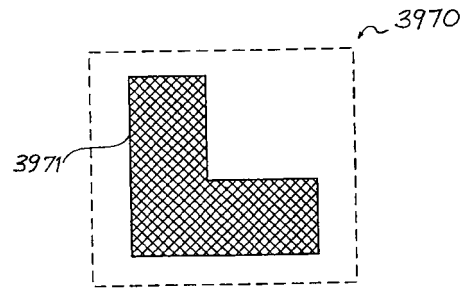
【図 2 7 4】



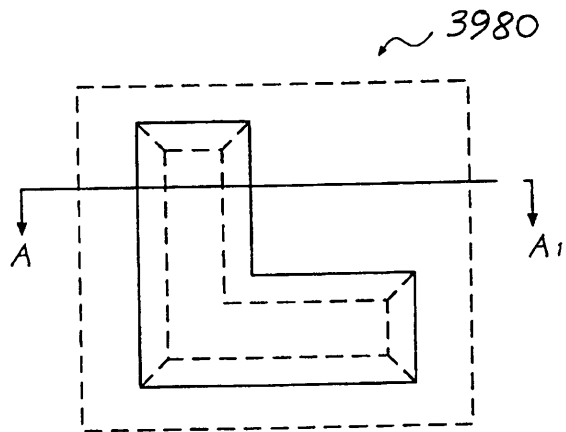
【図 2 7 5】



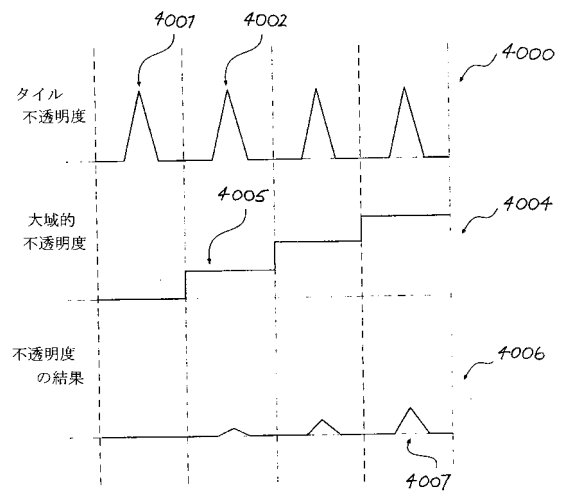
【図 2 7 6】



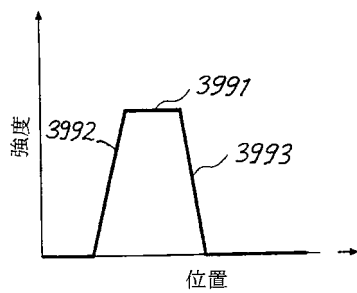
【図 2 7 7】



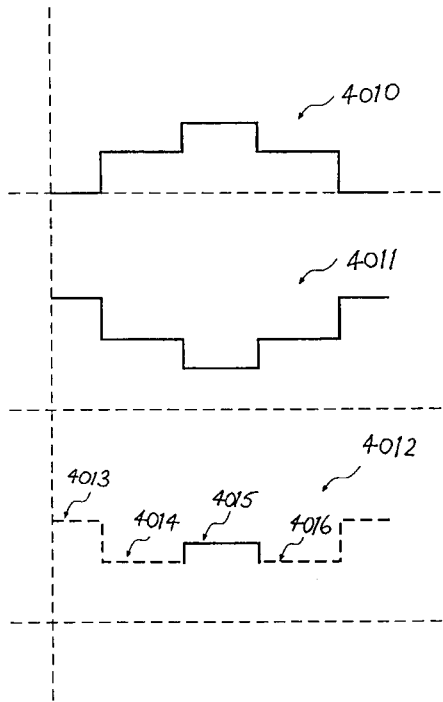
【図 2 7 9】



【図 2 7 8】



【図 280】

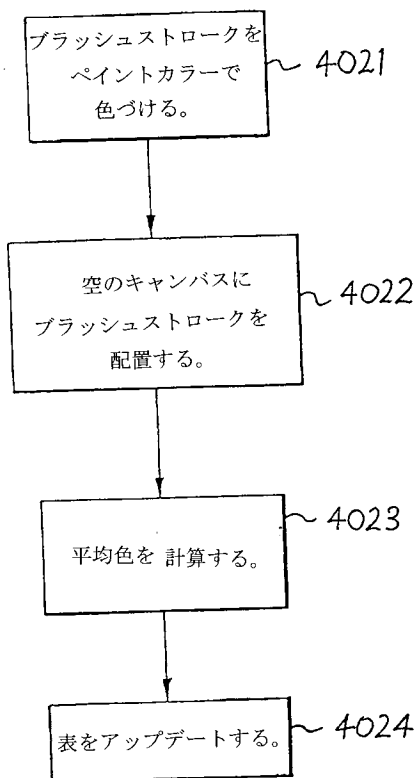


【図 281】

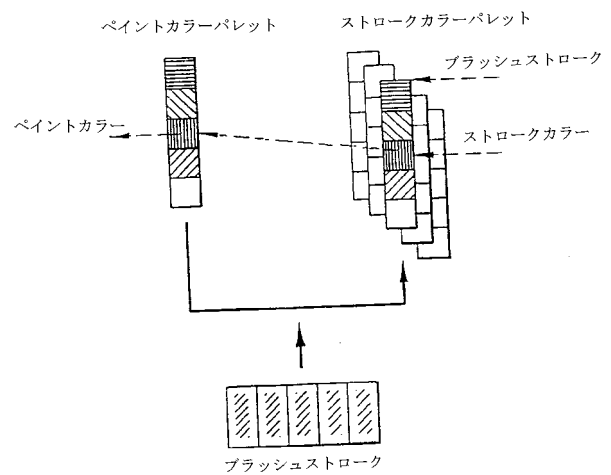
4025
↙

ブラッシュ ストローク	ペイント カラー	ストローク カラー
B_1	C_1 C_2 C_3 C_4	S_1 S_2 S_3 S_4
B_2	C_1 C_2 C_3 C_4	S_1 S_2 S_3 S_4
B_3	C_1 C_2 C_3 C_4	S_1 S_2 S_3 S_4
B_4	C_1 C_2 C_3	S_1 S_2 S_3

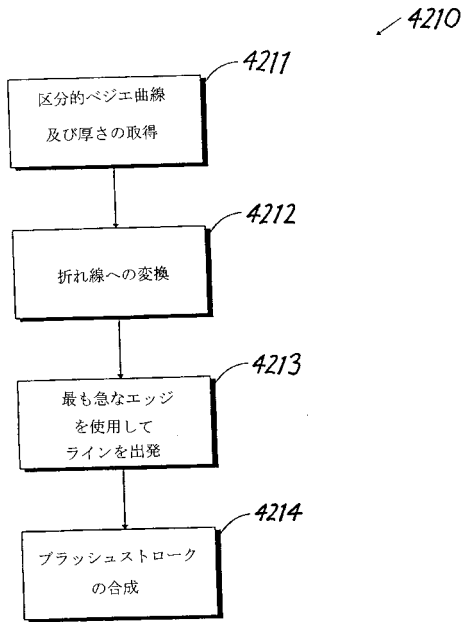
【図 282】



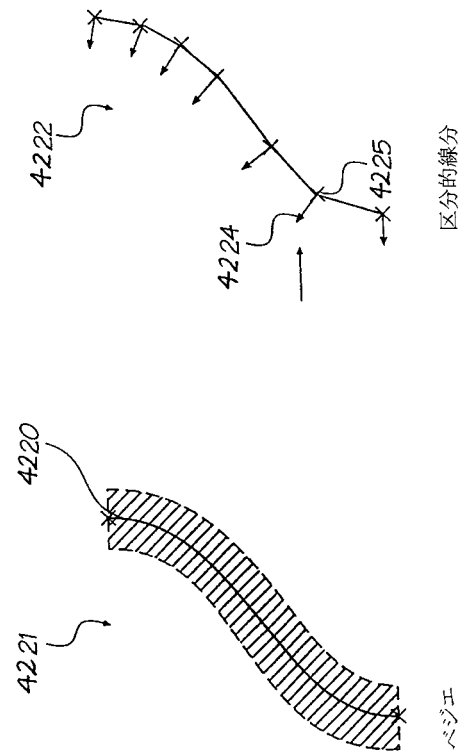
【図 283】



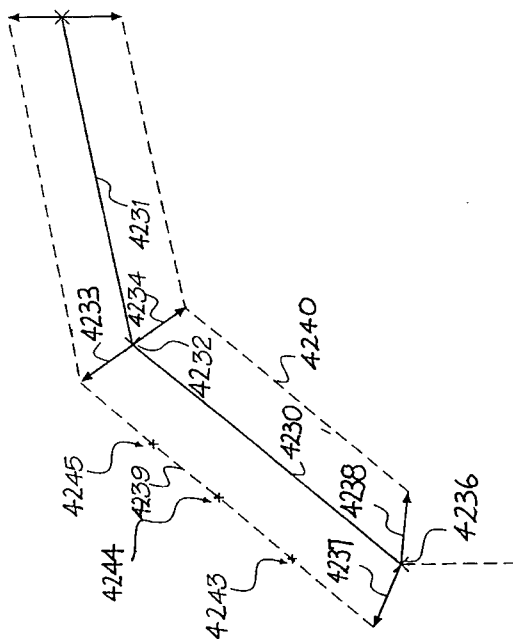
【図 284】



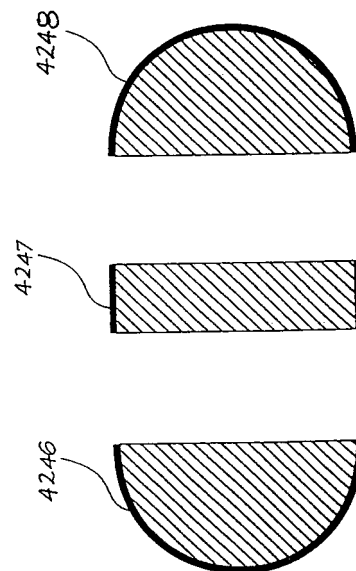
【図 285】



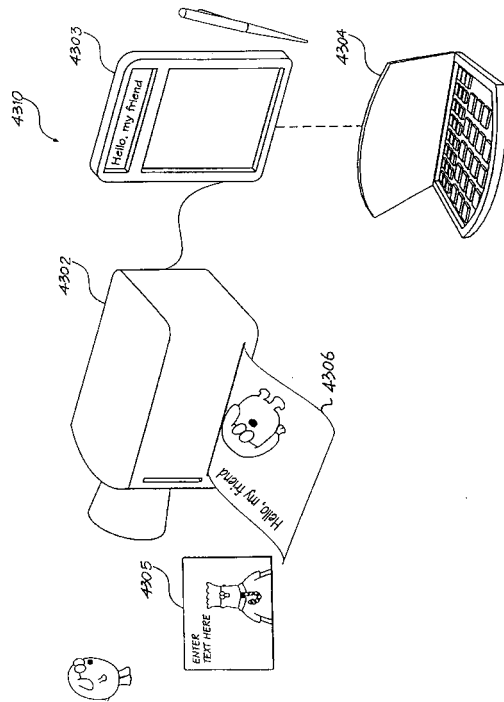
【図 286】



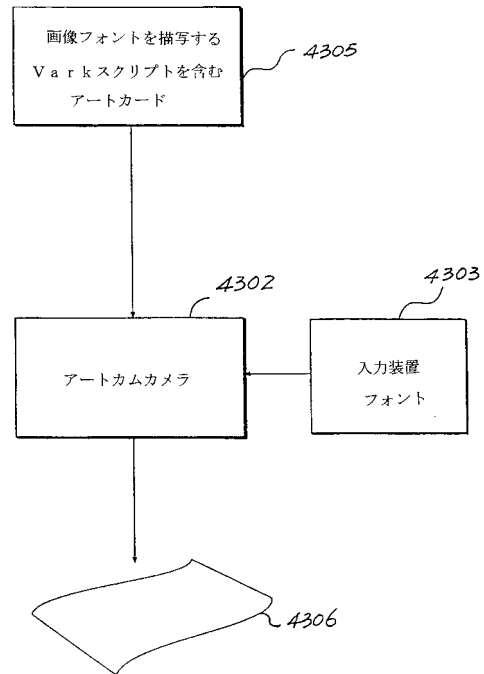
【図 287】



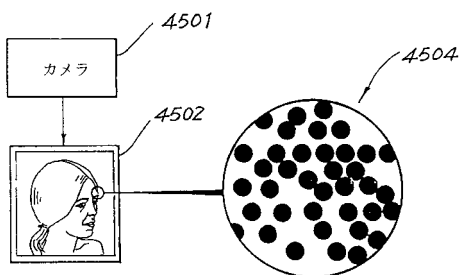
【図 288】



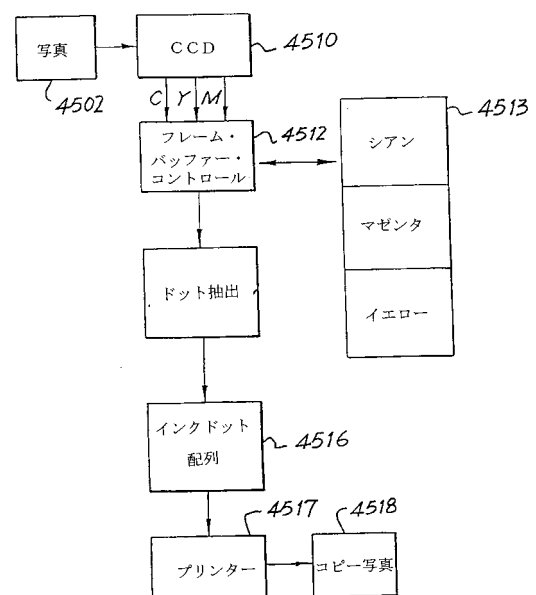
【図 289】



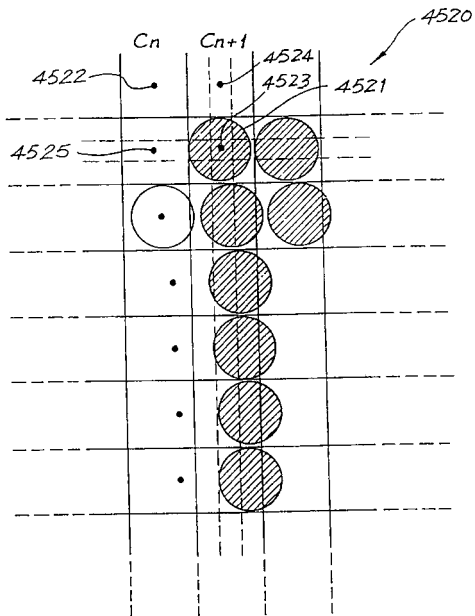
【図 290】



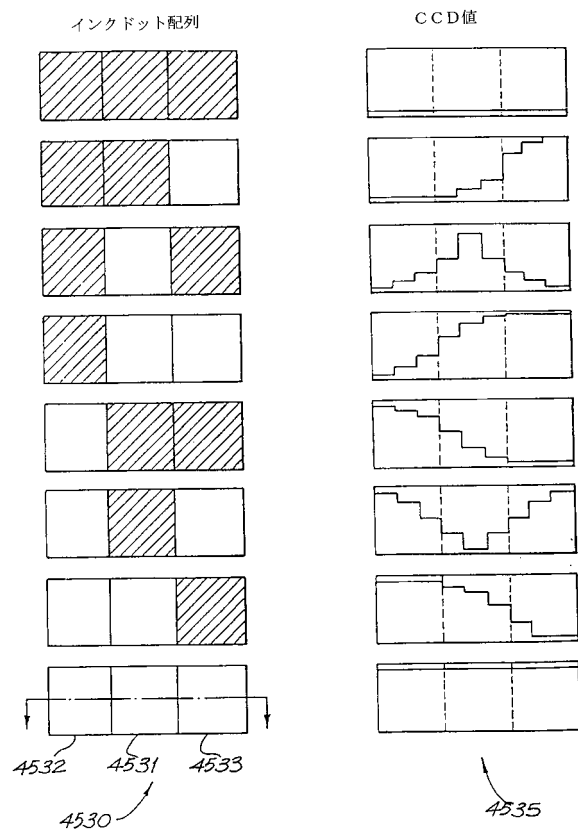
【図 291】



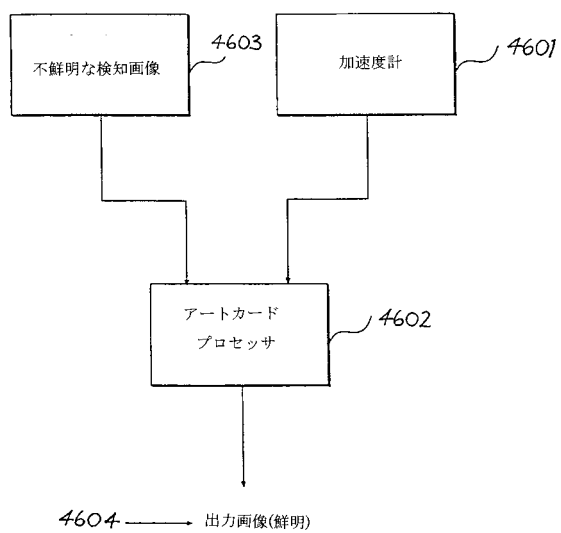
【図 2 9 2】



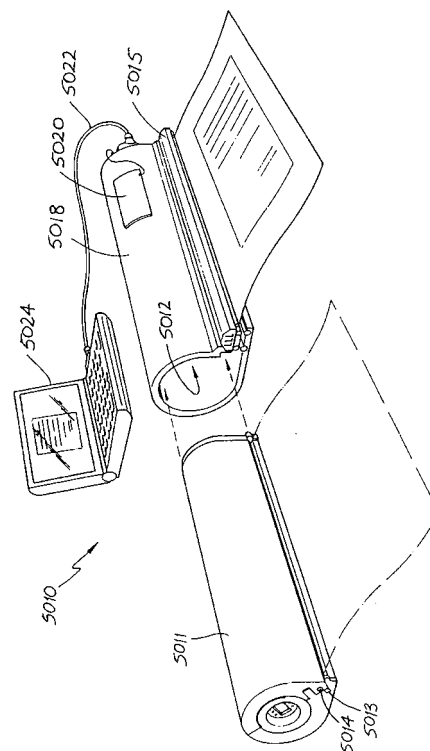
【図 2 9 3】



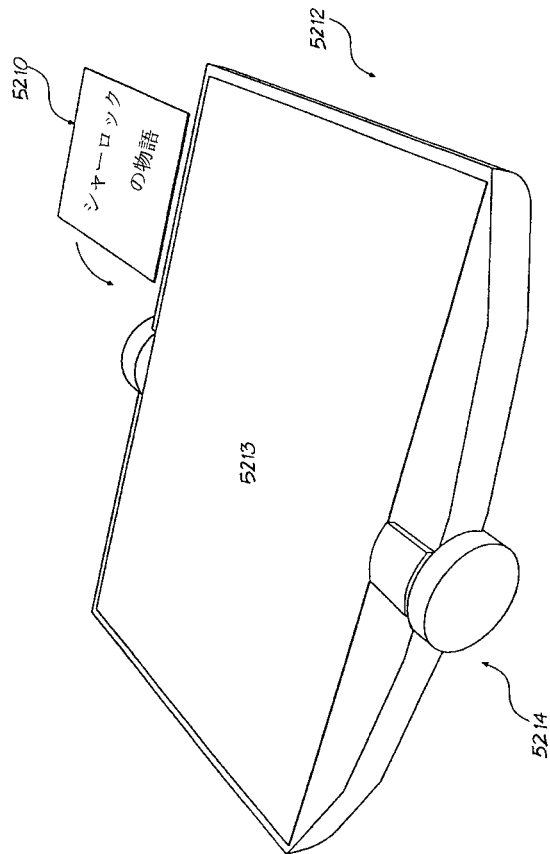
【図 2 9 4】



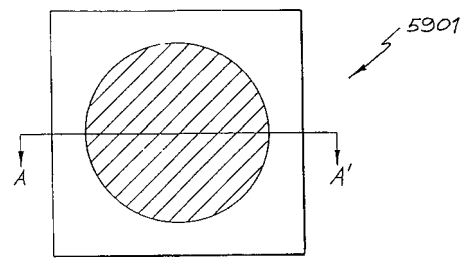
【図 2 9 5】



【図 2 9 6】



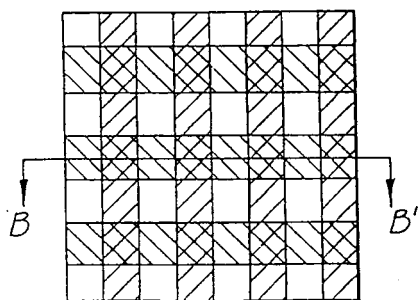
【図 2 9 7】



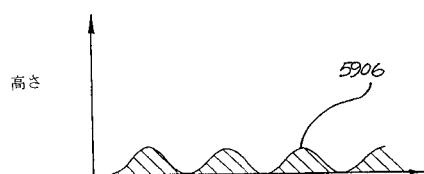
【図 2 9 8】



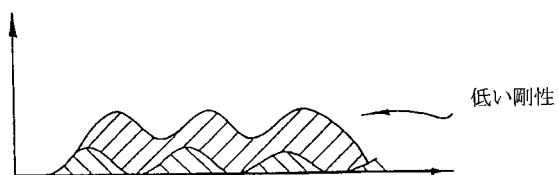
【図 2 9 9】



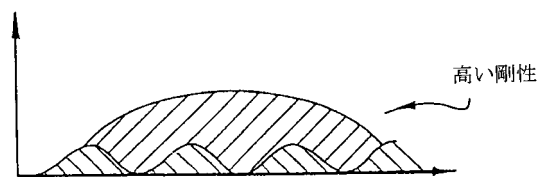
【図 3 0 0】



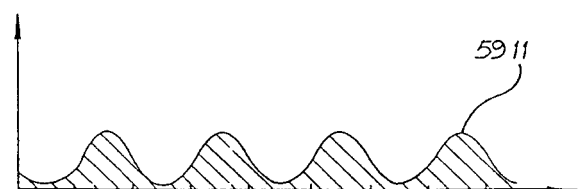
【図 3 0 1】



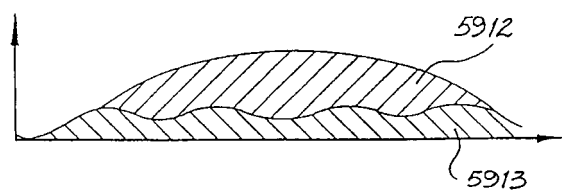
【図 3 0 2】



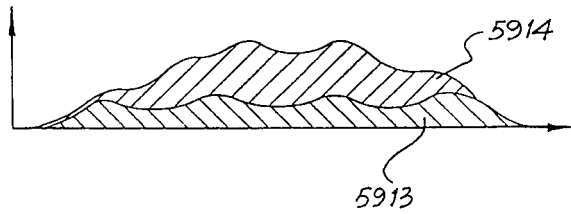
【図 3 0 3】



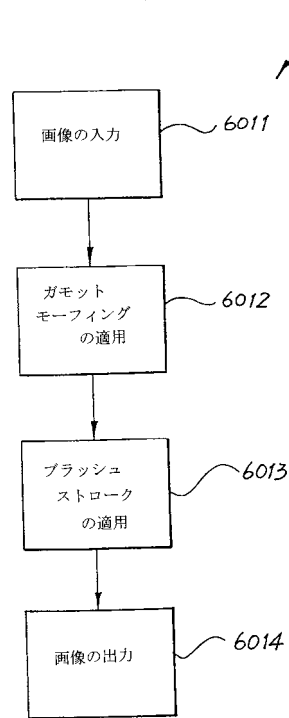
【図 3 0 4】



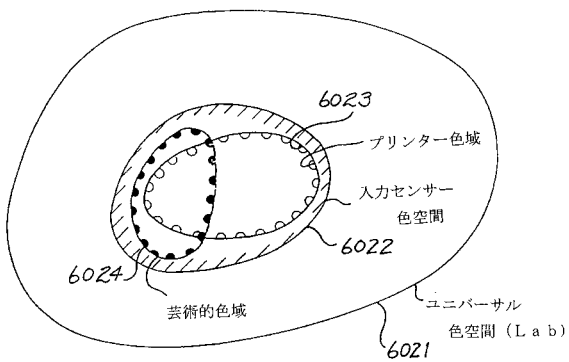
【図 3 0 5】



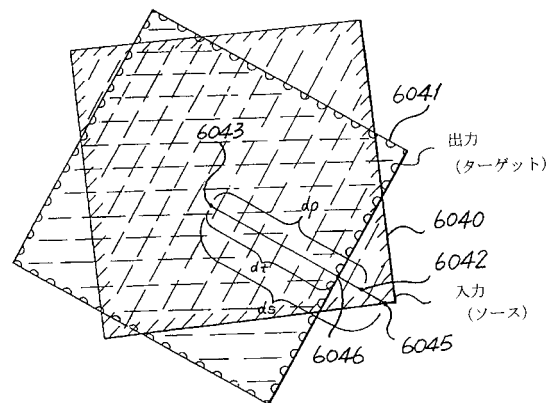
【図 3 0 6】



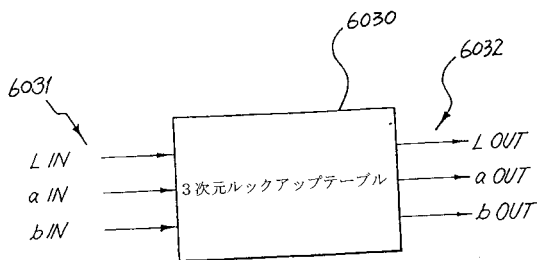
【図 3 0 7】



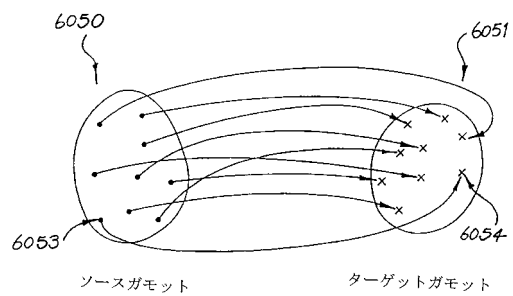
【図 3 0 9】



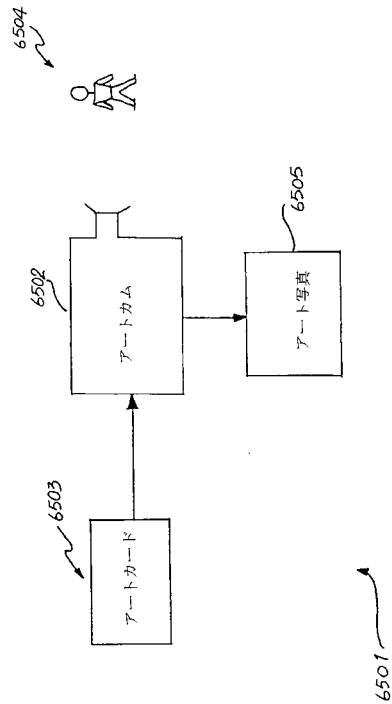
【図 3 0 8】



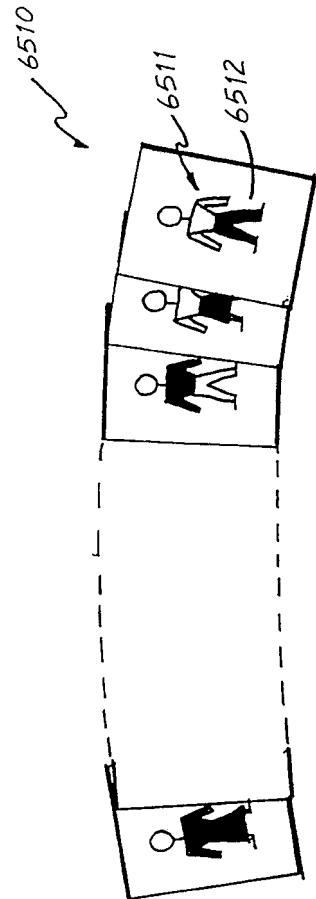
【図 3 1 0】



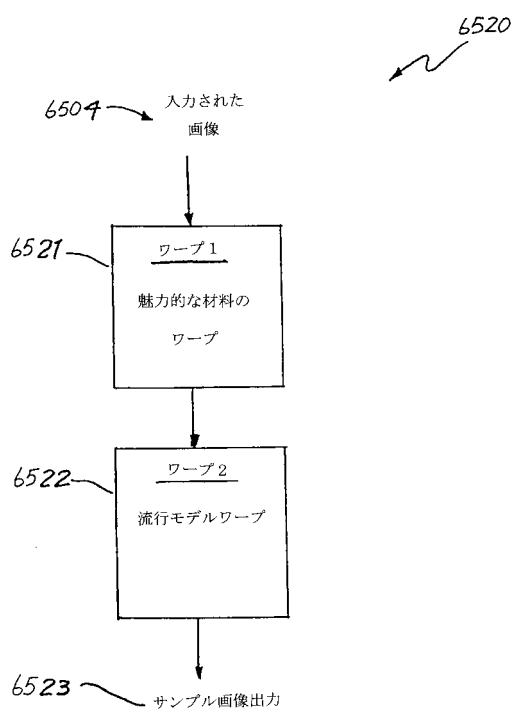
【図 3 1 1】



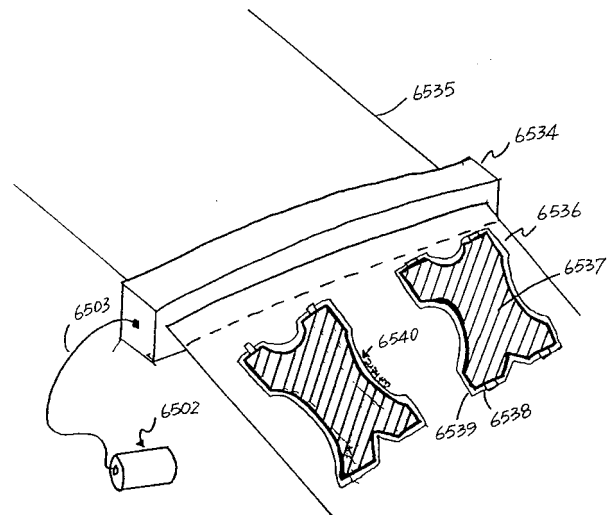
【図 3 1 2】



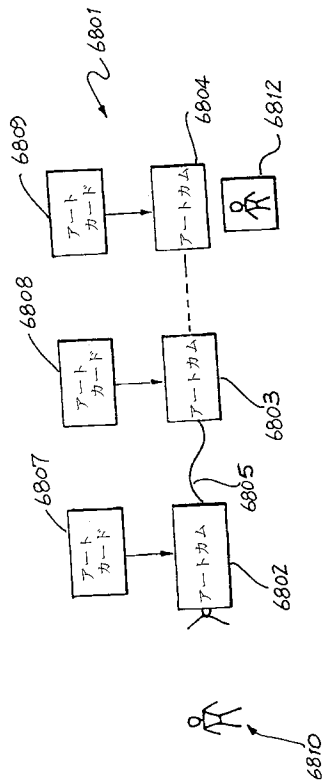
【図 3 1 3】



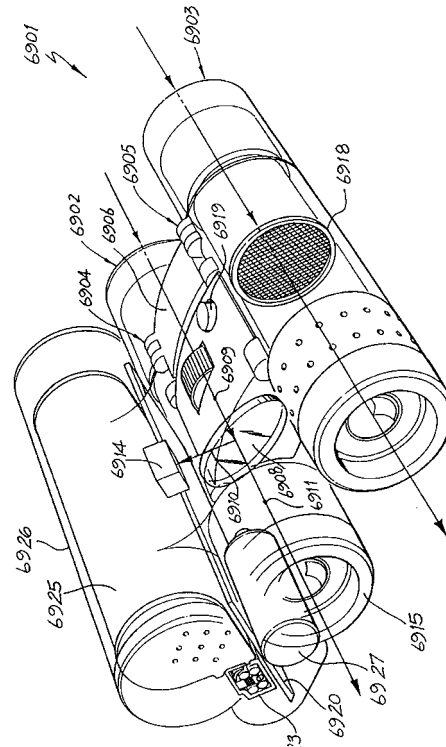
【図 3 1 4】



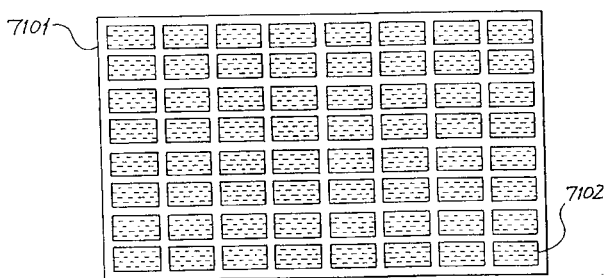
【図 3 1 5】



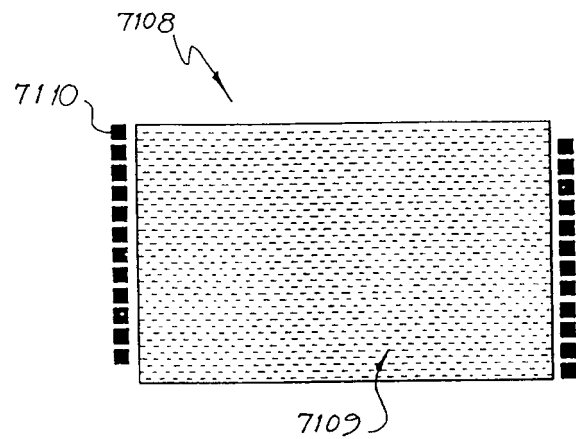
【図 3 1 6】



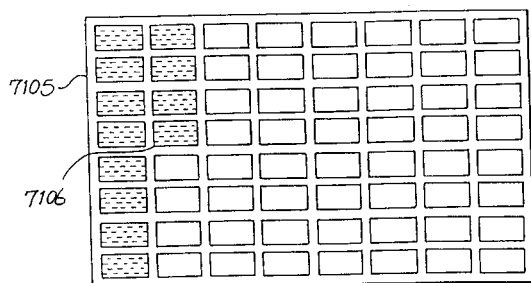
【図 3 1 7】



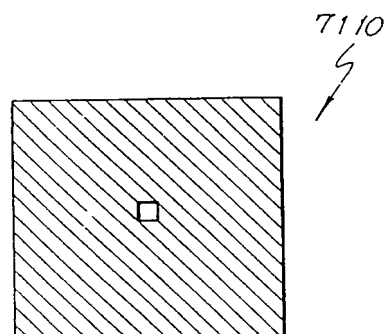
【図 3 1 9】



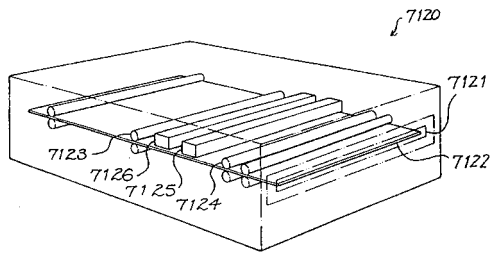
【図 3 1 8】



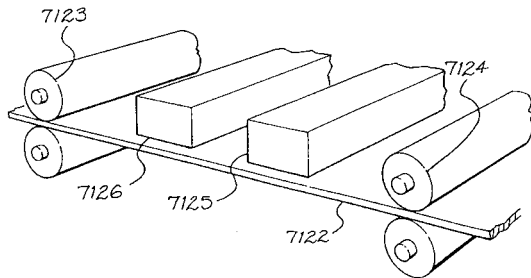
【図 3 2 0】



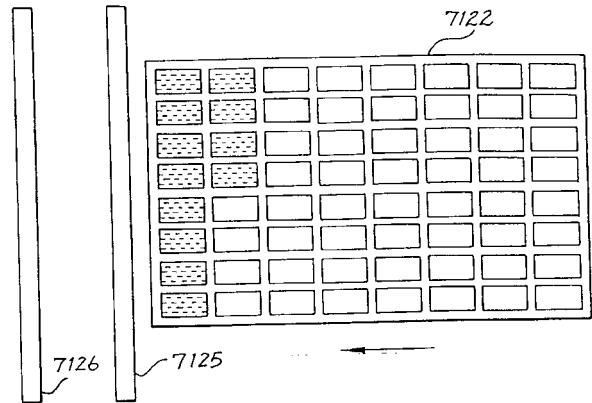
【図 3 2 1】



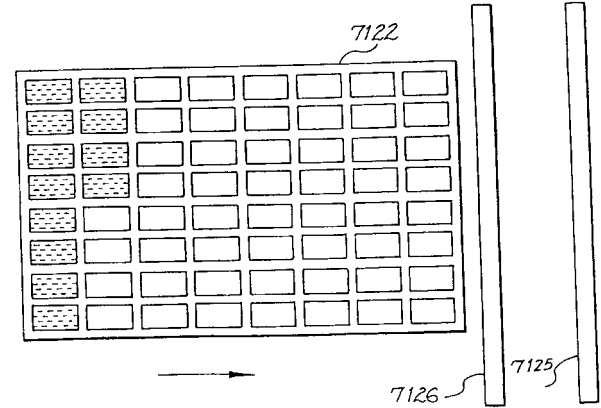
【図 3 2 2】



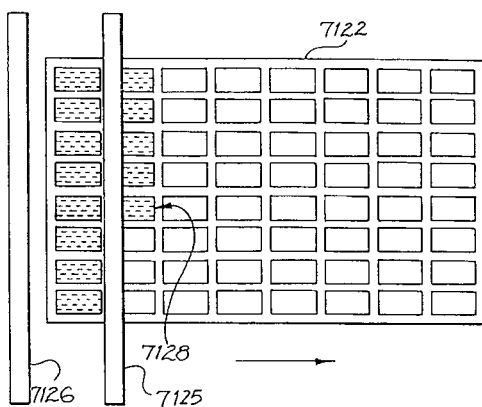
【図 3 2 3】



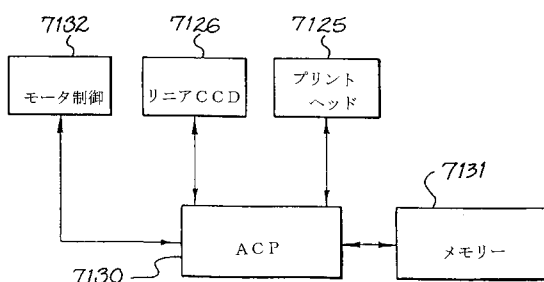
【図 3 2 4】



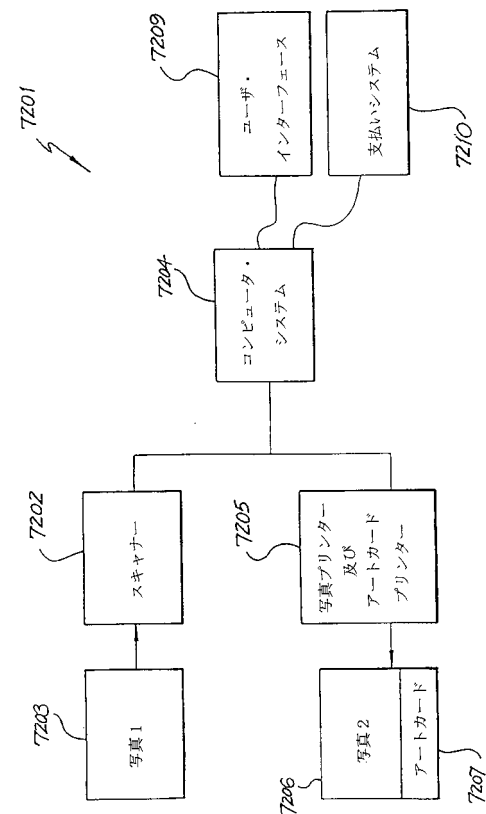
【図 3 2 5】



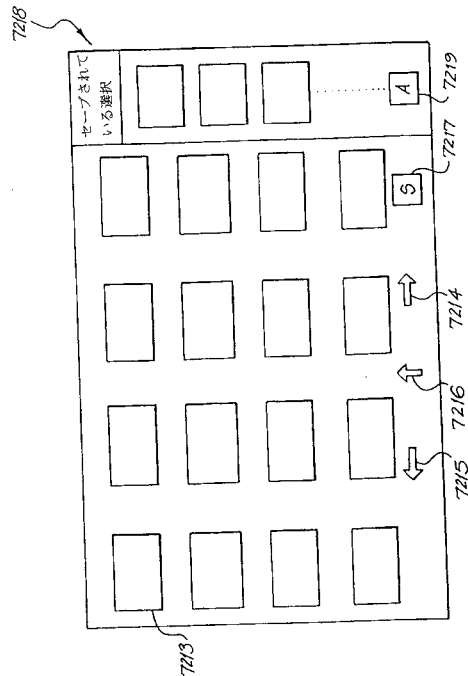
【図 3 2 6】



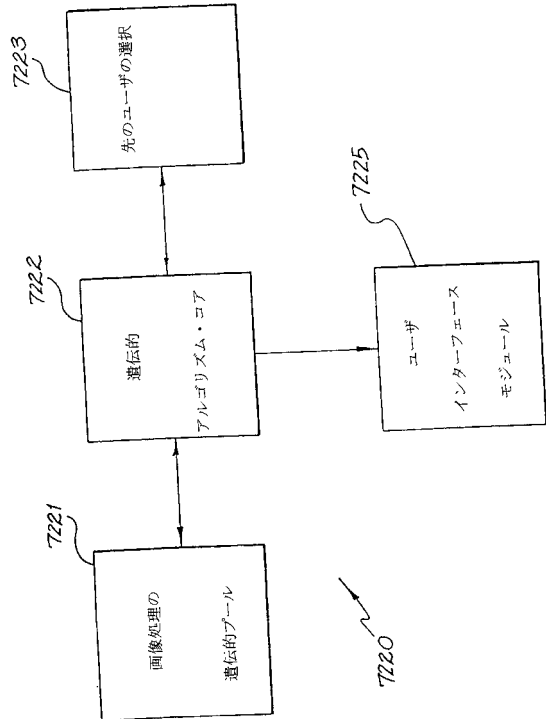
【図 3 2 7】



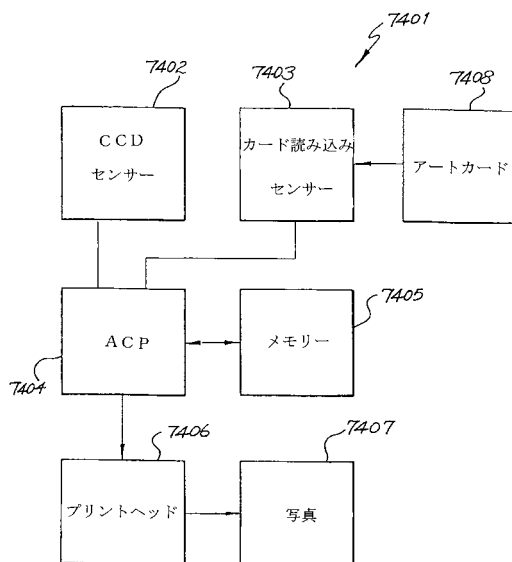
【図 3 2 8】



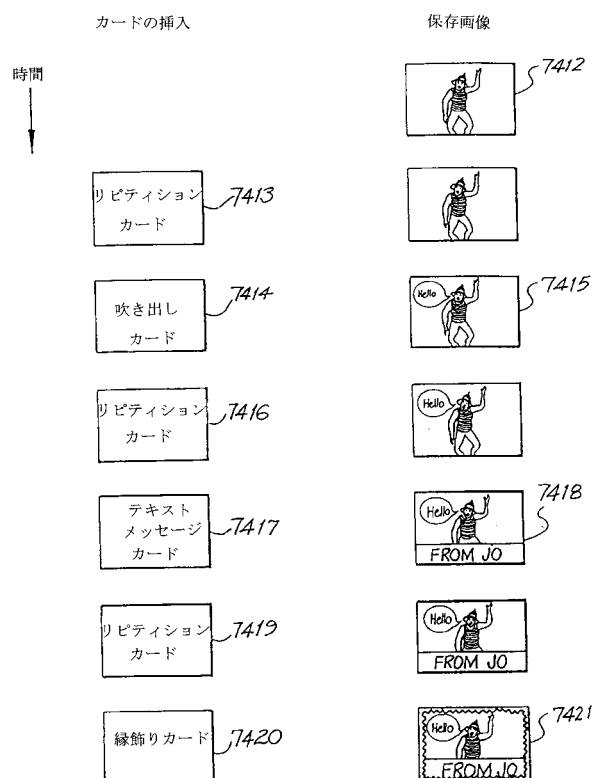
【図 3 2 9】



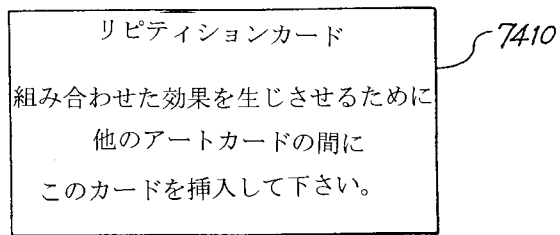
【図 3 3 0】



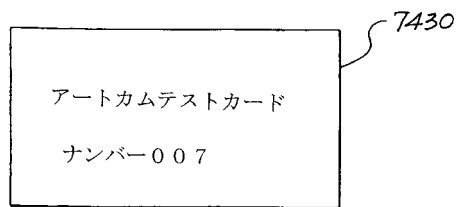
【図 3 3 1】



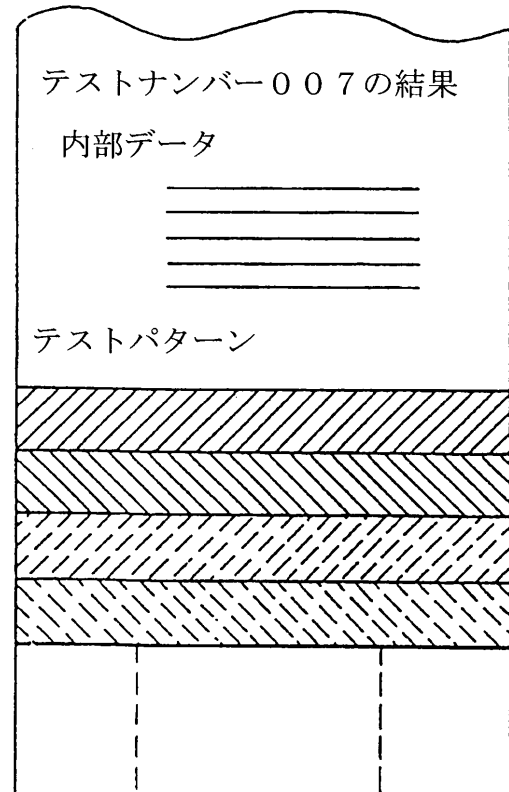
【図 3 3 2】



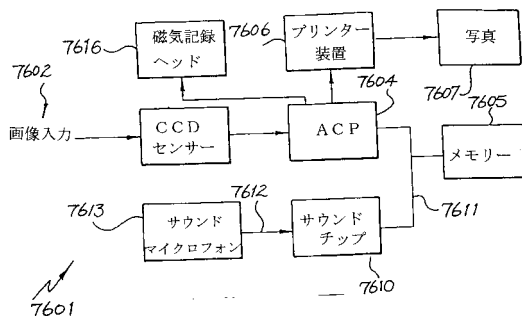
【図 3 3 3】



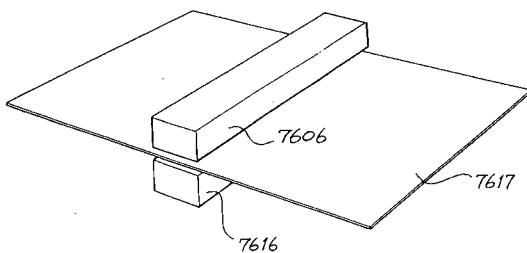
【図 3 3 4】



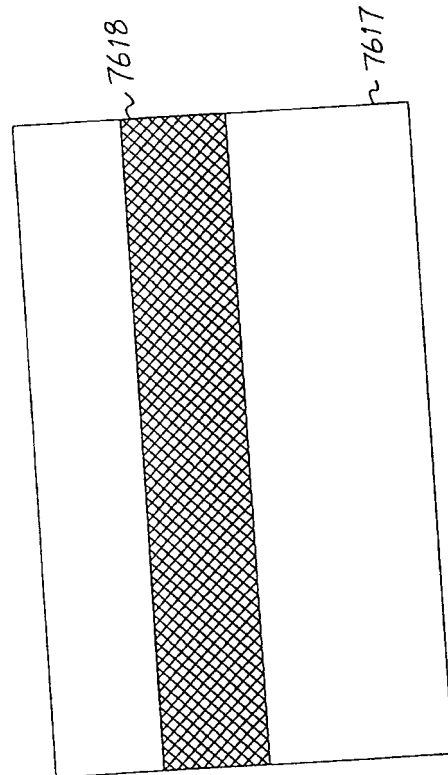
【図 3 3 5】



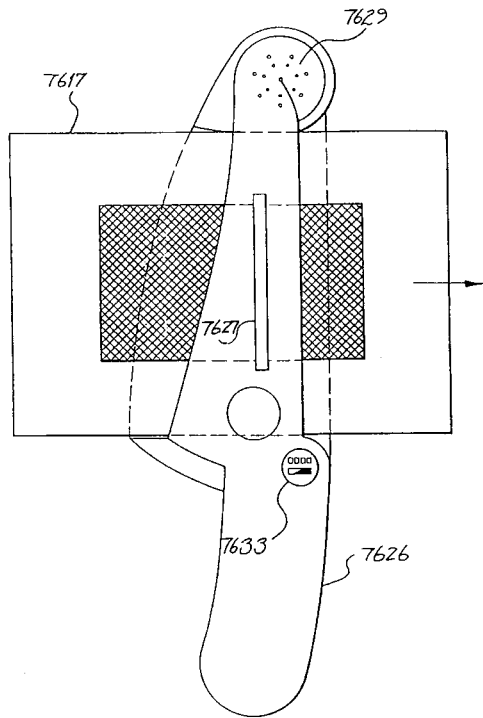
【図 3 3 6】



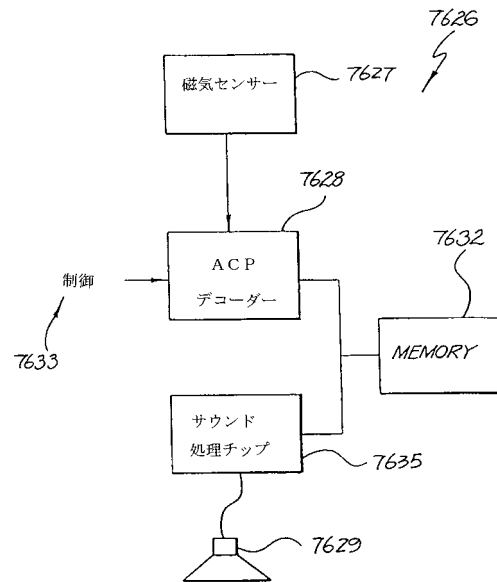
【図 3 3 7】



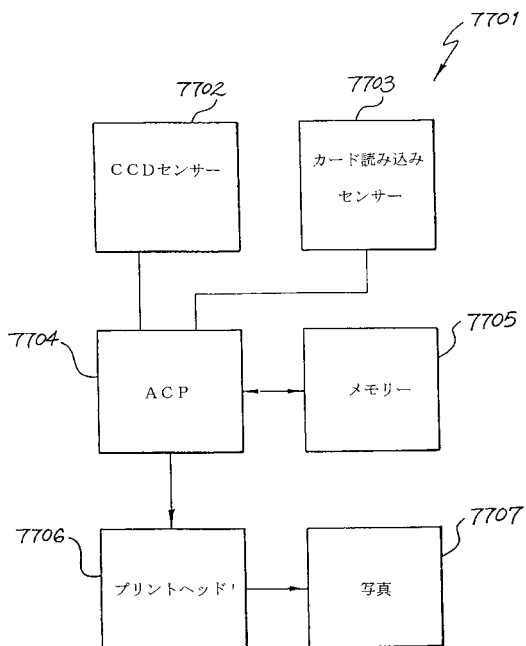
【図 3 3 8】



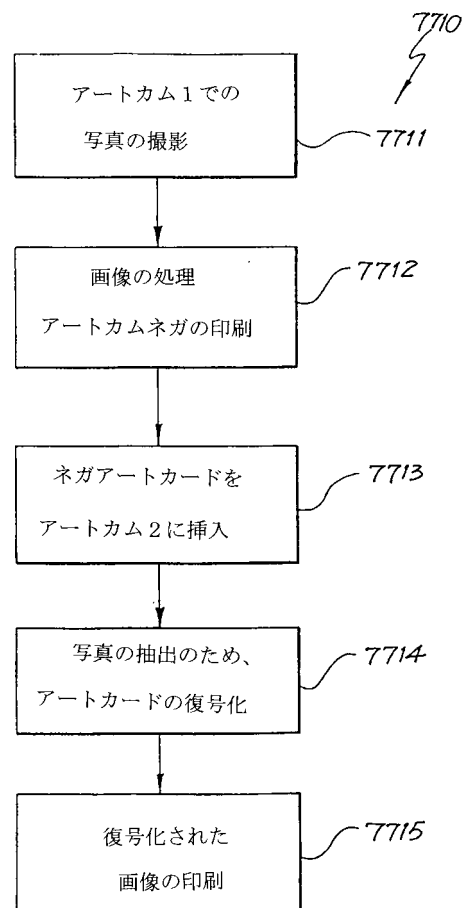
【図 3 3 9】



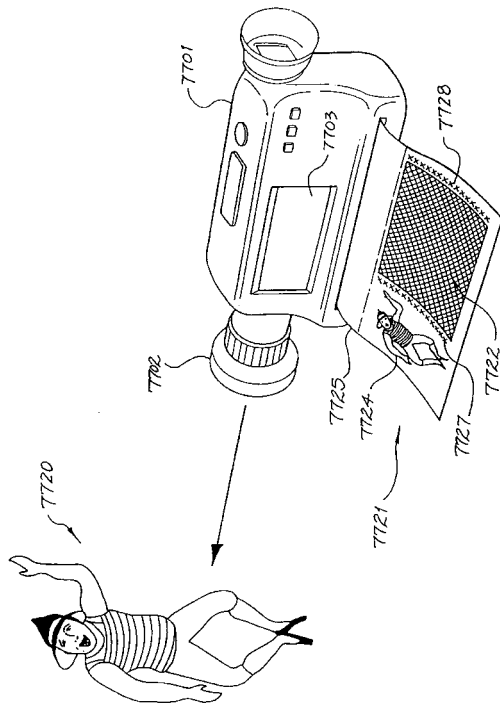
【図 3 4 0】



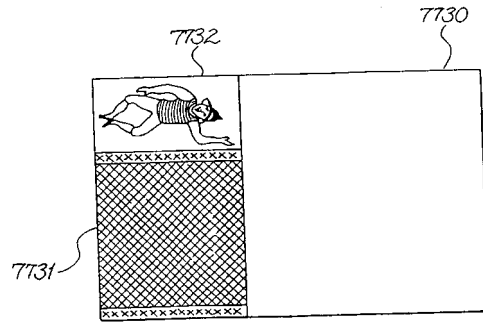
【図 3 4 1】



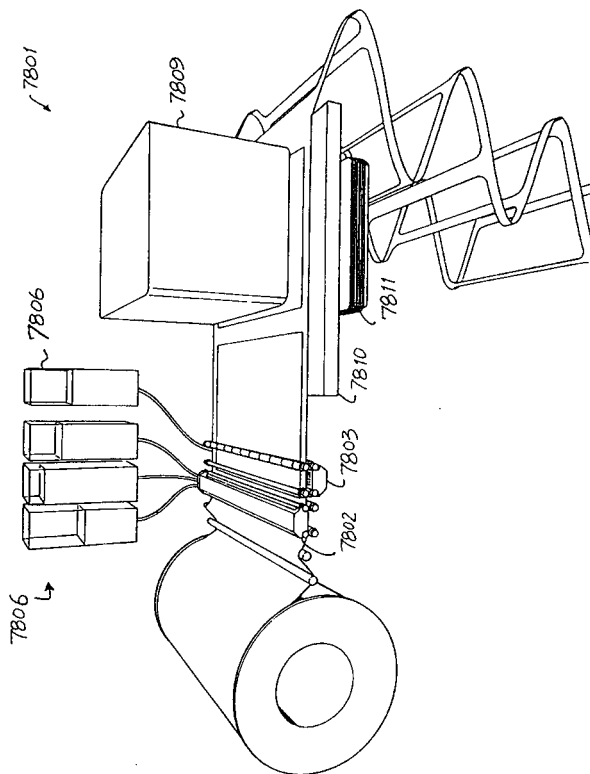
【図 3 4 2】



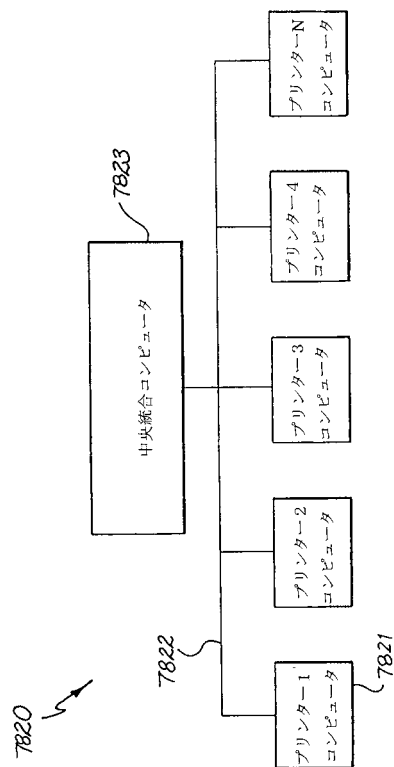
【図 3 4 3】



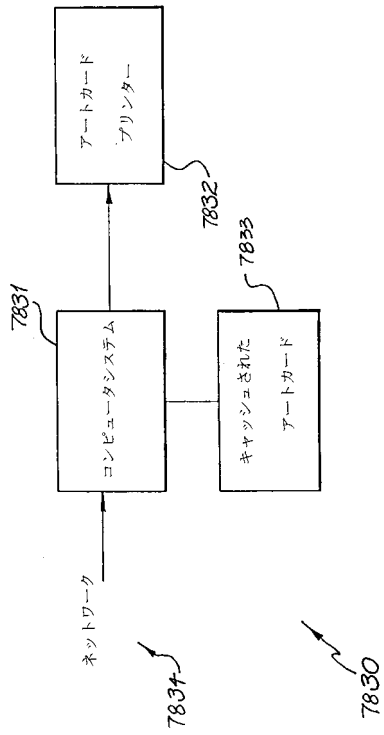
【図 3 4 4】



【図 3 4 5】



【図 3 4 6】



フロントページの続き

- (31)優先権主張番号 P07991
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07998
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07988
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07993
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08012
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08017
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08014
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08025
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08032
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07999
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08024
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08016
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08030
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07938
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07997
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07979
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)

- (31)優先権主張番号 P08015
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07978
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07982
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07989
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08019
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07980
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07942
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08018
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08021
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08000
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07940
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07939
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08020
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07985
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P07987
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08022
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
- (31)優先権主張番号 P08029
(32)優先日 平成9年7月15日(1997.7.15)

(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08023
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08028
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08027
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08026
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P07983
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P07986
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P07981
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P07977
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P07934
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P07990
(32)優先日 平成9年7月15日(1997.7.15)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08497
(32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08505
(32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08498
(32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08504
(32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08501
(32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08500
(32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08502

- (32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P08499
(32)優先日 平成9年8月11日(1997.8.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09395
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09404
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09394
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09396
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09397
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09398
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09399
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09400
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09401
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09402
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09403
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 P09405
(32)優先日 平成9年9月23日(1997.9.23)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 PP0959
(32)優先日 平成9年12月16日(1997.12.16)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 PP1397
(32)優先日 平成10年1月19日(1998.1.19)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 PP2370
(32)優先日 平成10年3月16日(1998.3.16)
(33)優先権主張国 オーストラリア(AU)

- (31)優先権主張番号 PP2371
(32)優先日 平成10年3月16日(1998.3.16)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 PP4094
(32)優先日 平成10年6月12日(1998.6.12)
(33)優先権主張国 オーストラリア(AU)

(特許庁注：以下のものは登録商標)

1. フロッピー

- (72)発明者 ウォームスレイ サイモン
オーストラリア エヌエスダブリュ 2 1 2 1 エピング ペムブローク ストリート 9 ユニ
ット 3
(72)発明者 ラプスタン ポール
オーストラリア エヌエスダブリュ 2 0 4 6 ロード ポイント デューク アベニュー 1 3
Fターム(参考) 5C122 DA04 EA42 FA04 FC01 FH10 FH14 FK12 FK28 FK37 FK41
GB01 HA03 HA64 HA76 HB01 HB02 HB09 HB10

【外国語明細書】

2009022020000001.pdf

2009022020000002.pdf

2009022020000003.pdf

2009022020000004.pdf