(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4451908号
(P4451908)
(45)発行日　平成22年4月14日(2010.4.14)　(24)登録日　平成22年2月5日(2010.2.5)

(51)Int.Cl.　F I
G06F 17/21 (2006.01)　G06F 17/21 570L

請求項の数 27 (全 57 頁)

(21)出願番号	特願2008-50694(P2008-50694)	(73)特許権者	503260918
(22)出願日	平成20年2月29日(2008.2.29)		アップル インコーポレイテッド
(62)分割の表示	特願2007-182633(P2007-182633)の分割		アメリカ合衆国 95014 カリフォルニア州 クパチーノ インフィニット ループ 1
原出願日	平成8年9月13日(1996.9.13)	(74)復代理人	100155284
(65)公開番号	特開2008-192163(P2008-192163A)		弁理士 井原 光雅
(43)公開日	平成20年8月21日(2008.8.21)	(74)代理人	100077481
審査請求日	平成20年2月29日(2008.2.29)		弁理士 谷 義一
審判番号	不服2008-32772(P2008-32772/J1)	(74)代理人	100088915
審判請求日	平成20年12月25日(2008.12.25)		弁理士 阿部 和夫
(31)優先権主張番号	08/527,438	(72)発明者	エドバーグ, ピーター, ケイ.
(32)優先日	平成7年9月13日(1995.9.13)		アメリカ合衆国 97403 オレゴン州 ユージン フェアマウント ブールバード 1892
(33)優先権主張国	米国(US)		
(31)優先権主張番号	08/527,827		
(32)優先日	平成7年9月13日(1995.9.13)		
(33)優先権主張国	米国(US)		

最終頁に続く

(54)【発明の名称】ユニコード・コンバータ

(57)【特許請求の範囲】
【請求項１】
ユニコード変換のためにコンピュータ上でソース文字列をターゲット文字列に変換する方法であって、前記コンピュータは、ソース文字列のターゲット文字列への変換を制御するコンバータ手段と、前記ソース文字列をスキャンして前記ソース文字列のテキスト要素を判定するスキャナ手段と、複数のマッピング・テーブルを格納するメモリ手段とを含み、前記方法は、
（ａ）バッファ手段が、第１の文字コード化を有するソース文字列を受け取ることと、
（ｂ）前記スキャナ手段が、前記バッファ手段が受け取った前記ソース文字列をテキスト要素に順次分割することであって、各テキスト要素は、前記ソース文字列の１または複数の文字を含み、前記テキスト要素の少なくとも１つは、前記ソース文字列の複数の文字を含むことと、
（ｃ）前記スキャナ手段が、前記分割（ｂ）の後または間に前記テキスト要素についての属性情報を得ることと、
（ｄ）前記コンバータ手段が、前記テキスト要素のそれぞれについて第２の文字コード化に関連付けられた変換コードをマッピング・テーブルでルックアップすることであって、前記テキスト要素のそれぞれについての変換コードを有するマッピング・テーブルでルックアップすること（ｄ）は、要求された変種を識別するオペレーションと、前記属性情報、前記要求された変種、および前記テキスト要素の長さに基づいて前記メモリ手段に格納された複数のマッピング・テーブルの１つを選択するオペレーションと、前記マッピン

グ・テーブルの選択された１つから変換コードをルックアップすることを含むことと、
（ｅ）前記コンバータ手段が、前記第２の文字コード化のターゲット文字列を形成するように前記テキスト要素の前記変換コードを結合することと
を備えることを特徴とする方法。
【請求項２】
請求項１に記載の方法であって、
前記変換コードは、前記第２の文字コード化中の１または複数の文字からなり、
前記得られた属性情報は、方向、クラス、優先順位、サブセット、コンテキスト、シメトリック・スワッピング状態の少なくともいずれか２つを含むことを特徴とする方法。
【請求項３】
請求項１に記載の方法であって、前記テキスト要素は、互いに隣接し、複数の文字を含む前記テキスト要素のそれぞれについて、前記文字は、前記ソース文字列において隣接することを特徴とする方法。
【請求項４】
請求項１に記載の方法であって、前記マッピング・テーブルは、レギュラ・マッピングとフォールバック・マッピングを含み、
前記ルックアップすること（ｄ）は、前記レギュラ・マッピングを使用して前記マッピング・テーブルが前記テキスト要素についての変換コードを含んでいないとき、前記フォールバック・マッピングを使用して前記テキスト要素のそれぞれについての変換コードを判定することを特徴とする方法。
【請求項５】
請求項１に記載の方法であって、前記文字のそれぞれは、関連する文字クラスを有し、
前記分割すること（ｂ）は、前記ソース文字列内の前記文字の文字クラスに少なくとも一部基づくことを特徴とする方法。
【請求項６】
請求項１に記載の方法であって、前記分割すること（ｂ）は、
（ｂ１）前記ソース文字列から次のソース文字を獲得することと、
（ｂ２）獲得した前記ソース文字を現在のテキスト要素に含めるべきか、あるいはまた、新たな次のテキスト要素として始めるかをスキャナ・テーブルに基づいて判定することと、
（ｂ３）前記判定すること（ｂ２）に従い、獲得した前記ソース文字を前記現在のテキスト要素または前記新たな次のテキスト要素に配置することと、
（ｂ４）前記ソース文字列がテキスト要素に完全に配置されるまで、（ｂ１）から（ｂ３）までを繰り返すことと
を備えることを特徴とする方法。
【請求項７】
請求項６に記載の方法であって、前記判定すること（ｂ２）は、
（ｉ）前記スキャナ・テーブルにおいて前記ソース文字に関連付けられた属性情報をルックアップすることであって、前記属性情報は少なくともクラス・インディケータを含むことと、
（ｉｉ）前記クラス・インディケータに基づき、獲得した前記ソース文字を前記現在のテキスト要素に含めるべきか、あるいはまた、新たな次のテキスト要素として始めるかを判定することと
を備えることを特徴とする方法。
【請求項８】
請求項６に記載の方法であって、前記判定すること（ｂ２）は、
（ｉ）前記スキャナ・テーブルにおいて前記ソース文字に関連付けられた属性情報をルックアップすることであって、前記属性情報は少なくともクラス・インディケータを含むことと、
（ｉｉ）複数の状態を有するステート・マシンを提供することであって、前記ステート

・マシンは、前記クラス・インディケータおよび前記ステート・マシンの現在の状態に基づき、獲得した前記ソース文字を前記現在のテキスト要素に含めるべきか、あるいはまた、新たな次のテキスト要素として始めるかを判定するのに使用されることと、
（ｉｉｉ）前記ステート・マシンの前記現在の状態を更新することと
を備えることを特徴とする方法。
【請求項９】
請求項１に記載の方法であって、前記第１の文字コード化および前記第２の文字コード化の１つは、ユニコード標準に準拠することを特徴とする方法。
【請求項１０】
請求項１に記載の方法であって、前記結合すること（ｅ）は、
（ｅ１）前記第２の文字コード化についてターゲット文字サイズを判定することと、
（ｅ２）ルックアップされた前記テキスト要素の前記変換コードを、前記ターゲット文字サイズの単位で前記ターゲット文字列にコピーすることにより前記ターゲット文字列を形成することと
を備えることを特徴とする方法。
【請求項１１】
請求項１０に記載の方法であって、ルックアップ（ｄ）された前記変換コードは、間接シーケンスまでのオフセットを指定することを特徴とする方法。
【請求項１２】
請求項１に記載の方法であって、
（ｆ）前記分割（ｂ）の後であるが前記ルックアップ（ｃ）の前に、前記コンバータ手段またはスキャナ手段が、ある文字が前記テキスト要素に存在する場合、前記テキスト要素のそれぞれの前記ある文字を再配列することをさらに備えることを特徴とする方法。
【請求項１３】
請求項１２に記載の方法であって、前記再配列すること（ｆ）は、異なる文字クラスの加重値を用いて実行されることを特徴とする方法。
【請求項１４】
ユニコードについてソース文字列をターゲット文字列に変換するコード変換システムであって、
第１の文字コード化を有する前記ソース文字列について第２の文字コード化を有する前記ターゲット文字列への変換を制御するコンバータと、
前記コンバータに動作するよう結合され、前記ソース文字列をテキスト要素に分割するスキャナであって、各テキスト要素は前記ソース文字列の１または複数の文字を含み、前記テキスト要素の少なくとも１つは前記ソース文字列の複数の文字を含むスキャナと、
前記ソース・コード化のテキスト要素についてターゲット・コード化を格納するマッピング・テーブルであって、複数のマッピング部分を含み、前記テキスト要素の属性情報および長さに応じて前記マッピング部分の適切な１つが前記テキスト要素のそれぞれに利用されるマッピング・テーブルと、
前記コンバータおよび前記マッピング・テーブルに動作するよう結合され、前記テキスト要素のそれぞれについて第２の文字コード化に関連付けられた変換コードを前記マッピング・テーブルでルックアップするルックアップ・ハンドラと
を備えたことを特徴とするコード変換システム。
【請求項１５】
請求項１４に記載のコード変換システムであって、
前記コンバータに動作するように結合され、特定の場合にフォールバック変換コードを提供するフォールバック・ハンドラであって、前記ルックアップ・ハンドラが１または複数のテキスト要素について変換コードを提供できないとき、前記フォールバック変換コードは、前記テキスト要素の文字と等価ではないが、類似するグラフィカル上の外観を有する、前記ターゲット・コード化における１または複数のコード・ポイントを含む、フォールバック・ハンドラをさらに備えたことを特徴とするコード変換システム。

【請求項１６】
請求項１５に記載のコード変換システムであって、
前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素として始めるかの判定において前記スキャナを支援するスキャナ・テーブル手段をさらに備えたことを特徴とするコード変換システム。
【請求項１７】
請求項１４に記載のコード変換システムであって、
前記スキャナに動作するよう結合され、前記入力文字列中の個々の文字を現在のテキスト要素内に含めるべきか、あるいはまた、新たな次のテキスト要素として始めるかの判定において前記スキャナを支援するスキャナ・テーブルをさらに備えたことを特徴とするコード変換システム。
【請求項１８】
請求項１７に記載のコード変換システムであって、前記ソース文字列の前記文字は、関連する文字クラスを有し、
前記スキャナ・テーブルは、要素の配列を備え、前記配列は文字クラスによりインデックス付けされていることを特徴とするコード変換システム。
【請求項１９】
請求項１４に記載のコード変換システムであって、前記ソース文字列中の前記文字は、ユニコード文字であることを特徴とするコード変換システム。
【請求項２０】
請求項１４に記載のコード変換システムであって、前記ターゲット文字列中の前記文字は、ユニコード文字であることを特徴とするコード変換システム。
【請求項２１】
ユニコード変換のためにコンピュータ上でソース文字列をターゲット文字列に変換する方法であって、前記コンピュータは、ソース文字列のターゲット文字列への変換を制御するコンバータ手段と、前記ソース文字列をスキャンして前記ソース文字列のテキスト要素を判定するスキャナ手段と、複数のマッピング・テーブルを格納するメモリ手段とを含み、前記方法は、
（ａ）バッファ手段が、第１の文字コード化を有するソース文字列を受け取ることと、
（ｂ）前記スキャナ手段が、前記バッファが受け取った前記ソース文字列をテキスト要素に順次分割することであって、各テキスト要素は、前記ソース文字列の１または複数の文字を含み、前記テキスト要素の少なくとも１つは、前記ソース文字列の複数の文字を含むことと、
（ｃ）前記スキャナ手段が、前記分割（ｂ）の後または間に前記テキスト要素についての属性情報を得ることと、
（ｄ）前記コンバータ手段が、前記テキスト要素のそれぞれについて第２の文字コード化に関連付けられた変換コードをマッピング・テーブルでルックアップすることであって、前記テキスト要素のそれぞれについての変換コードを有するマッピング・テーブルでルックアップすること（ｄ）は、（ｄ１）前記属性情報、および前記テキスト要素の長さに基づいて前記メモリ手段に格納された複数のマッピング・テーブルの１つを選択するオペレーションと、（ｄ２）前記マッピング・テーブルの選択された１つから変換コードをルックアップするオペレーションを含むことと、
（ｅ）前記コンバータ手段が、前記第２の文字コード化のターゲット文字列を形成するように前記テキスト要素の前記変換コードを結合することと
を備えることを特徴とする方法。
【請求項２２】
請求項２１に記載の方法であって、前記分割すること（ｂ）は、
（ｂ１）前記ソース文字列から次のソース文字を獲得することと、
（ｂ２）獲得した前記ソース文字を現在のテキスト要素に含めるべきか、あるいはまた、新たな次のテキスト要素として始めるかをスキャナ・テーブルに基づいて判定すること

と、
（ｂ３）前記判定すること（ｂ２）に従い、獲得した前記ソース文字を前記現在のテキスト要素または前記新たな次のテキスト要素に配置することと、
（ｂ４）前記ソース文字列がテキスト要素に完全に配置されるまで、（ｂ１）から（ｂ３）までを繰り返すことと
を備えることを特徴とする方法。
【請求項２３】
請求項２２に記載の方法であって、前記判定すること（ｂ２）は、
（ｉ）前記スキャナ・テーブルにおいて前記ソース文字に関連付けられた属性情報をルックアップすることであって、前記属性情報は少なくともクラス・インディケータを含むことと、
（ｉｉ）複数の状態を有するステート・マシンを提供することであって、前記ステート・マシンは、前記クラス・インディケータおよび前記ステート・マシンの現在の状態に基づき、獲得した前記ソース文字を前記現在のテキスト要素に含めるべきか、あるいはまた、新たな次のテキスト要素として始めるかを判定するのに使用されることと、
（ｉｉｉ）前記ステート・マシンの前記現在の状態を更新することと
を備えることを特徴とする方法。
【請求項２４】
請求項２３に記載の方法であって、
（ｆ）前記分割（ｂ）の後であるが前記ルックアップ（ｃ）の前に、前記コンバータ手段またはスキャナ手段が、ある文字が前記テキスト要素に存在する場合、前記テキスト要素のそれぞれの前記ある文字を再配列することをさらに備えることを特徴とする方法。
【請求項２５】
請求項１４に記載のコード変換システムであって、
前記マッピング・テーブルは、前記ソース・コード化のあるテキスト要素について１つ以上のターゲット・コード化を含み、前記あるテキスト要素についての前記ターゲット・コード化の特定の１つを前記あるテキスト要素について判定された属性情報によって獲得することを特徴とするコード変換システム。
【請求項２６】
請求項１４に記載のコード変換システムであって、
前記コード変換システムは、
スキャン情報を格納するスキャナ・テーブルと、
前記ソース文字列中の文字についての属性情報を格納する属性テーブルと
をさらに備え、
前記スキャナは、前記スキャナ・テーブルに格納されたスキャナ情報および前記属性テーブルに格納された属性情報に従って前記ソース文字列をテキスト要素に分割するように動作することを特徴とするコード変換システム。
【請求項２７】
請求項２１に記載の方法であって、
前記テキスト要素のそれぞれについて変換コードのマッピング・テーブルでルックアップすること（ｄ）は、（ｄ３）前記選択（ｄ１）の前に要求された変種を特定するオペレーションをさらに含み、
前記複数のマッピング・テーブルの１つを選択すること（ｄ２）は、前記属性情報、前記要求された変種、および前記テキスト要素の長さに基づき行われることを特徴とする方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は記述または表示されるテキストの文字コード間で変換を行うシステムに関し、さらに具体的には、ある文字セットと別の文字セット間で変換を行うコード・コンバータ

に関する。
【背景技術】
【０００２】
コンピュータや他のエレクトロニック・デバイスはテキストを使用してユーザと対話するのが一般的である。テキストはモニタや他のタイプの表示デバイスから表示されるのが普通である。テキストはコンピュータや他のエレクトロニック・デバイスの内部ではディジタル形式で表現しなければならないので、文字セット・コード化を使用しなければならない。一般的には、文字セット・コード化は文字セットの各文字を固有のディジタル表現でコード化するように働く。文字（コード化される）は文字、数字および種々のテキスト・シンボルに対応し、コンピュータや他のエレクトロニック・デバイスで使用される数値コードが割り当てられている。コンピュータや他のエレクトロニック・デバイスで使用されている最もよく知られた文字セットは情報交換用米国標準コード(American National Standard Code for Information Exchange - ＡＳＣＩＩ)である。ＡＳＣＩＩはそのコード化のために７ビット・シーケンスを使用している。他の国では、異なる文字セットが使用されている。ヨーロッパでは、支配的な文字コード化標準は国際標準化機構(International Standards Organization - ＩＳＯ）によって開発されたＩＳＯ　８８５９－Ｘ、特にＩＳＯ　８８５９－１（「Ｌａｔｉｎ－１」と呼ばれている）である。日本では、支配的な文字コード化標準はＪＩＳ　Ｘ０２０８であり、ここでＪＩＳとは日本情報標準(Japanese Information Standard)のことであり、日本標準協会（Japan Standards Association - ＪＳＡ）によって開発されたものである。その他の既存文字セットの例としては、Ｍａｃ（登録商標）ＯＳ　Ｓｔａｎｄａｒｄ　Ｒｏｍａｎコード化（Apple Computer, Inc 開発）、Ｓｈｉｆｔ－ＪＩＳ（日本）、Ｂｉｇ５（台湾）、その他多くがある。
【発明の開示】
【発明が解決しようとする課題】
【０００３】
ビジネスとネットワークのグローバル化に伴い、コンピュータや他のエレクトロニック・デバイスが複数の文字コード化を処理する能力をもつことが重要になっている。例えば、同じコンピュータやエレクトロニック・デバイスは、コンピュータや他のエレクトロニック・デバイスと異なる言語で対話することを望んでいる異なる国籍の人達で使用されることがある。このような各言語では、異なる文字セット・コード化が必要になるのが通常である。しかし、文字セットは同じ言語の場合でも、異なることがある。
【０００４】
もう１つの要求は、ある文字セット・コードから別の文字セット・コード化への変換ができることである。例えば、ＩＳＯ　８８５９－１を使用するフランスのユーザは、電子メール・メッセージをＩＳＯ　８８５９－８を使用するイスラエルのユーザにフランス語で送りたい場合がある。送信者と受信者は異なる文字セット・コード化を使用しているので、メッセージの中の非ＡＳＣＩＩ文字は、イスラエルのユーザには誤って伝えられることになる。理想的なことは、コンピュータまたはエレクトロニック・デバイスの１つがある文字セットから別の文字セットに変換することである。このことは、限られた範囲でいくつかの文字セット間ですでに達成されているが、最新のコンピュータやエレクトロニック・デバイスでもほとんど不可能である。コード変換を困難にしているのは、異なる文字標準が多数存在し、各国の標準が矛盾し、または不統一であることがよくあるためである。
【０００５】
Ｕｎｉｃｏｄｅ（登録商標）標準（以下では、単にユニコードまたはユニコード標準という）は国際的文字コード化標準となるために開発されたものである。ユニコード標準の設計者はより効率的で、柔軟性のある文字識別方法を必要として、提供したものである。ユニコード標準には、１９９０年１２月３１以前に承認され、公表されたすべての主要国際標準のすべての文字が含まれている他に、以前の標準になかった他の文字も含まれている。これらの文字は重複することなくユニコード標準にコード化されている。ユニコード

標準に含まれるコードは１６ビット（または２バイト）幅である。

【０００６】

ユニコード標準などの文字セット標準はコード変換を容易にし、テキスト・データに働きかける有用なプロセスのインプリメンテーションを可能にしている。例えば、上記の例によれば、フランス国内のコンピュータや他のエレクトロニック・デバイスはユニコード文字を送信することができ、イスラエル国内のコンピュータや他のエレクトロニック・デバイスは受信したユニコード文字を、イスラエル国内のコンピュータや他のエレクトロニック・デバイスと互換性のあるヘブライ語の文字に変換することができる。

【０００７】

以下では、ユニコード標準の概要について説明する。なお、ユニコード標準の詳細は、例えば、「The Unicode Standard, Worldwide Character Encoding, Version 1.0, Addision-Wesley 1991」に記載されている（バージョン１．１にも詳細が記載されている）。これらのどちらのバージョンも、その全体は引用により本明細書の一部を構成するものである。

【０００８】

ユニコード・コード化方式の設計は方向性を除き、基本的テキスト処理アルゴリズムの設計から独立している。ユニコード・インプリメンテーションは適当なテキスト処理および／またはレンダリング・アルゴリズムを含んでいるものと想定されている。便宜上、ユニコード標準のすべてのコードは言語および機能のカテゴリ別に分類されているが、ユニコード標準のすべてのコードは等しくアクセス可能になっている。「ユニコード標準のコード・スペースは６つのゾーンに分割されている。すなわち、一般的スクリプト（アルファベットおよび相対的に小さな文字セットをもつ他のスクリプト）、シンボル（記号）、ＣＪＫ（中国語、日本語および朝鮮語）補足文字、ＣＪＫ表意文字、プライベート使用および互換性である。一般的スクリプト・ゾーンはラテン、キリル、ギリシア、ヘブライ、アラビア、デーヴァナーガリーおよびタイなどのアルファベットまたは音節スクリプトを含んでいる。シンボル・ゾーンは句読点、数学、化学、読者の注意をひく記号などの、様々な種類の文字を含んでいる。ＣＪＫ補足ゾーンは句読点、シンボル、カナ文字、Ｂｏｐｏｍｏｆｏ文字、および単一および複合ハングル文字を含んでいる。ＣＪＫ表意文字ゾーンには、２０，０００個以上の表意文字または他のスクリプトからの文字のためのスペースが用意されている。プライベート使用ゾーンはユーザまたはベンダ固有のグラフィック文字を定義するために使用される。互換性ゾーンはユニコード・コード化で他の規範的表現をもつ、幅広く使用されている企業および国内標準からの文字を含んでいる。」（上記標準バージョン１．０の１３ページから引用）。

【０００９】

ユニコード標準は文字特性と制御文字を含んでいる。文字特性はユニコード・コード化に含まれるコード・ポイントに関する意味論的知識を必要とする解析、ソート、および他のアルゴリズムで利用すると便利である。ユニコード標準で指定されている文字特性には、ディジット、数字、スペース文字、非スペース・マーク、および方向がある。ユニコード文字は文字特性に基づいてグループ化されている。ディジット、数字およびスペース文字は周知である。非スペース・マーク・グループは非スペース・マークを収容し、方向グループは方向文字を収容している。

【００１０】

非スペース・マーク（例えば、ギリシアとローマ・スクリプトにおけるアクセント・マーク、アラビアとデーヴァナーガリーにおける母音マーク）は最終的にレンダリングされたテキストには線形的に現れない。ユニコード文字コード・シーケンスでは、このような文字はすべて、その文字で修正されたベース文字、またはその後に音声順（phonetic order）に分節される文字のあとに置かれている（例えば、ローマ文字の“Á”は事前構成形式（precomposed form）でストアされないとき“Ａ´”としてストアされる）。これらの文字（つまり、非スペース・マーク）はレンダリングされるとき、その前に置かれたベース文字に対してなんらかの方法で位置付けられることを目的とし、これらの文字自体はスペース位置を占めない。制御文字はユニコード標準でコード化されているが、制御文字自体はグラフィック文字ではない。これらの制御文字は、例えば、水平タブを示したり、フォーマッティング属性または構造などのテキストに関する追加情報を提供したり、方向性フォーマッティングによって制御したりするために使用される。ユニコード標準は双方向文字配列も用意しているので、ユニコード・コード化方式には、方向の変更を指定する文字も含まれている。例えば、ギリシア語、ローマ語およびタイ語では、左から右への方向が支配的になっているのに対し、アラビア語とヘブライ語では、右から左への方向が支配的になっている。制御文字を使用して方向を変更することは、例えば、左から右への文字を右から左への順序で表示したいときに、ユーザまたはシステムによって要求されることがある。

【００１１】

従来のコード・コンバータには、１つの問題がある。それは、コンバータが１つのソース文字しか１つのターゲット文字に変換できないことである。この種の変換はある種の文字セットでは有効に働くが、多数の非スペース文字セットを含むある種の文字セット（例えば、ユニコード）の場合や、通常他の文字と関連づけられていないマークを結合するときに十分ではない。また、従来のコード・コンバータは複数の文字に関連するシンボル、合字(ligature)または表意文字との間で変換できる機能を備えていない。

【００１２】

その結果として、従来のコード・コンバータにはラウンドトリップ忠実度(round trip fidelity)がない。ラウンドトリップ忠実度とは、コード・コンバータが変換し、そのあと変換を戻してオリジナルの入力文字列を再現できる能力のことである。これが重要になるのは、コード・コンバータをある文字コードから別の文字コードに変換するときのハブとして使用するときである。

【００１３】

従って、複数のソース文字から単一ターゲット文字へ、あるいは単一ソース文字から複数のターゲット文字へ変換して、ラウンドトリップ文字コード変換の忠実度を保証するようなコード・コンバータが望まれている。

【００１４】

従来のコード・コンバータには、もう１つ問題がある。それは、ソース文字セットの文字をターゲット文字セットの文字に変換するとき方向を考慮に入れないことである。方向を考慮に入れないと、ある文字は左から右へ配列されるのに対し、他の文字は右から左へ配列されるので変換にエラーを生じることになる。これが起こる代表例として、ある与えられたソース・コード化に対して２つの等価文字をもつターゲット文字に変換し、違いが方向だけのときである。このような場合には、正しいターゲット文字をマッピングするために、ソース文字の方向が分かっていなければならない。また、従来のコード・コンバータが不十分であるのは、左から右へ配列されている（方向性）言語だけでなく、右から左

へ配列されている言語にも対応する文字を含んでいるある種の文字セット（例えば、ユニコード）を取り扱えるだけの柔軟性がないためである。例えば、文字の配列または方向性がユニコード文字列内で変化するようなユニコード文字は、従来のコード・コンバータが文字列全体の方向が一定であることを前提としているので従来のコード・コンバータでは正しく変換されないことになる。

【００１５】

従って、文字の方向を考慮に入れながら、ソース文字セットの文字をターゲット文字セットの文字に変換することができるようなコード・コンバータが望まれている。

【００１６】

ある種の従来のコード・コンバータには、以下のようなとき起こる別の問題がある。ネットワークからテキストを受信するとき、そのテキストに関連するデータはデータ・ブロックで到着するのが代表的である。データが完全な形で受信されるのは、データを構成するすべてのブロックが受信されたあとだけである。受信データはバッファに置かれ、そこでデータは文字コードの変換を待っている。しかし、バッファは、多くの場合、データ・ストリーム全体を格納する能力もなければ、その１ブロックさえも格納する能力をもっていないこともある。いずれの場合も、スキャナ４０８によって後でテキスト要素と判断されるはずのものの途中でバッファの終わり（つまり、バッファ内の最後の文字）が現れることがある。テキスト要素の途中でバッファの終わりが現れると、スキャナ４０８は、後続文字が最後のテキスト要素に影響したり、その一部であることがあるので最後のテキスト要素を正しく得ることができなくなる。

【００１７】

大部分のコード・コンバータには、別の問題がある。それは、あるソース文字セットの文字をあるターゲット文字セットの文字に変換するときコンテキストを考慮に入れないことである。ある種の文字セット（例えば、ユニコード）では、文字セットにはコンテキスト表現形式が異なるごとに別々の文字コードが含まれているのに対し、別の時では、単一文字コードだけが用意され、コンテキストは表現形式を判断するために使用されている。しかし、従来のコード・コンバータはコードをそのコンテキストに従って変換することができない。コンテキスト・ベースのコード変換が特に問題となるのは、変換の対象となる文字セット（例えば、ユニコード）が文字コンテキストを受け入れるために、いくつかの手法を組み合わせて利用するときである。

【００１８】

従って、文字のコンテキストを考慮に入れながら、ソース文字セットの文字をターゲット文字セットの文字に正確かつ柔軟に変換できるようなコード・コンバータが望まれている。

【課題を解決するための手段】

【００１９】

概要を説明すると、本発明はコード変換および／または切り捨て処理手法に関するものである。

【００２０】

本発明の第１側面では、コード変換手法によれば、ラウンドトリップ忠実度が得られると共に、その結果の文字コードが他のプラットフォームと互換性をもつことが保証される。コード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする能力をもっている。ラウンドトリップ忠実度によれば、ソース・テキストはターゲット・テキストに変換し、その後オリジナル・ソース・テキストに再び戻るように変換することが可能である。互換性は標準ターゲット文字の使用を最大限にし、プライベート文字の使用を最小限にすることにより保証される。コード変換は他の文字セットからユニコード文字へ、ユニコード文字から他の文字セットへ変換するとき利用すると、特に便利である。ユニコード文字シーケンスをターゲット文字セット内の単一文字にマッピングすることは、従来は提供されていなかった。本発明は、オペレーションだけでなく、データ保管においても大幅な柔軟性が得られる

強力な解決法を提供している。本発明は様々な態様で実現することができる。つまり、方法、装置またはシステムとして実現することもコンピュータ可読メディア上に実現することも可能である。

【００２１】

ソース文字列をターゲット文字列に変換する方法としては、本発明の第１側面による本発明の実施例は、第１文字コード化をもつソース文字列を受信し、ソース文字列をテキスト要素に順次に分割し（各テキスト要素はソース文字列の１つまたは複数の文字を含んでいる）、テキスト要素の各々ごとに第２文字コード化に関連する変換コードをマッピング・テーブルで調べ（ルックアップ）、テキスト要素の変換コードを結合して第２文字コード化のターゲット文字列を形成するオペレーションを実行する。

【００２２】

さらに、本発明の第１側面によれば、マッピング・テーブルにはレギュラ・マッピングとフォールバック・マッピングを含めることが可能であり、ルックアップ・オペレーションは、レギュラ・マッピングを使用するテキスト要素の変換コードがマッピング・テーブルにないとき、フォールバック・マッピングを使用するテキスト要素の各々の変換コードを判断することが可能である。また、好ましいことは、文字の各々にそれぞれに関連する文字クラスをもたせ、少なくともその一部がソース文字列内の文字の文字クラスに基づいてソース文字列の分割が行われるようにすることである。

【００２３】

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第１側面による本発明の実施例は、第１文字コード化をもつソース文字列を第２文字コード化をもつターゲット文字列に変換することを制御するコンバータ、ソース文字列をテキスト要素（各テキスト要素はソース文字列の１つまたは複数の文字を含んでいる）に分割するスキャナ、ソース・コード化のテキスト要素に対するターゲット・コード化をストアしておくマッピング・テーブルおよびテキスト要素の各々ごとに第２文字コード化に関連する変換コードをマッピング・テーブルで調べる（ルックアップ）ルックアップ・テーブルとを含んでいる。

【００２４】

本発明の第１側面によるコード変換システムには、さらに、フォールバック・ハンドラとスキャナ・テーブルを含めることが可能である。フォールバック・ハンドラは、ルックアップ・ハンドラが１つまたは複数のテキスト要素の変換コードを提供することができないような、特殊な場合にフォールバック変換コードを提供する。フォールバック変換コードはテキスト要素内の文字と正確には等価ではないが、類似したグラフィック外観をもつターゲット・コード化内の１つまたは複数のコード・ポイントを含んでいる。スキャナ・テーブルは、入力文字列内の個別文字を現在のテキスト要素内に含めるべきか、それとも新しい次のテキスト要素を開始すべきかをスキャナが判断するのを支援するものである。

【００２５】

入力文字列をスキャンするスキャニング・システムとしては、本発明の第１側面による本発明の実施例は、入力文字を入力文字列から取得する入力デバイス（入力文字列は文字コード化をもち、入力文字列の各文字は文字クラスをもっている）、入力文字の属性を提供する属性テーブルおよびステート・マシンの次の状態と次のアクションの両方を、入力文字の次の状態とステート・マシンの現在状態に従って判断するステート・マシンを含んでいる。スキャニング・システムは入力文字列の入力文字を現在のテキスト要素内に含めるべきか、現在のテキスト要素を終わらせて次のテキスト要素を開始させるかを、ステート・マシンが判断したアクションに基づいて判断する。好ましくは、属性には、入力文字の文字クラスが含まれおり、ステート・マシンは入力文字の文字クラスとステート・マシンの現在状態に基づいて次の状態と次のアクションを判断する。

【００２６】

ソース文字列をターゲット文字列に変換するためのプログラム命令を収めておくコンピュータ可読メディアとしては、本発明の第１側面による本発明の実施例は、第１文字コー

ド化をもつソース文字列を受信することをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列をテキスト要素（各テキスト要素はソース文字列の１つまたは複数の文字を含んでいる）に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード、テキスト要素の各々ごとに第２文字コード化に関連する変換コードを調べる（ルックアップ）ことをコンピュータに実行させるように構成されたコンピュータ可読コード、およびテキスト要素の変換コードを結合して第２文字コード化のターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

【００２７】

本発明の第２側面では、コード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するとき方向を考慮に入れる。

【００２８】

本発明の第２側面によるコード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする能力を備えている。変換される文字の方向を判断または解明することにより、コード変換システムは判断または解明された文字の方向を利用して、ターゲット文字・コード化への正しいマッピングが利用されることを保証する。従って、本発明によれば、ソース文字列内の文字の方向性が変化したときでも正しいコード変換が達成される。本発明は様々な態様で実現することが可能である。方法、装置またはシステムとして実現することも、コンピュータ可読メディア上に実現することも可能である。

【００２９】

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第２側面による本発明の実施例は、ソース文字コード化をもつ入力文字列をターゲット文字コード化をもつターゲット文字列（入力文字列は複数の文字を含んでいる）に変換することを制御するコンバータ、入力文字列内の文字を判断するスキャナ、ソース・コード化の文字に対するターゲット・コード化をストアしておくマッピング・テーブル、および入力文字列内の文字の各々ごとにターゲット文字コード化に関連する変換コードを、入力文字列内の文字の方向とソース・コード化に基づいてマッピング・テーブルで調べるルックアップ・ハンドラを含んでいる。好ましくは、スキャナはさらに入力文字列をテキスト要素（各テキスト要素は入力文字列の１つまたは複数の文字を含んでいる）に分割し、スキャナはテキスト要素の方向を判断し、マッピング・テーブルはソース・コード化に対するターゲット・コード化をストアしており、ルックアップ・ハンドラはテキスト要素の各々ごとにターゲット文字コード化を、テキスト要素内の文字の方向とソース・コード化に基づいて調べる。この実施例には、前記ルックアップ・ハンドラが１つまたは複数の変換コードを提供できないような、特定な場合にはフォールバック変換コードを提供するフォールバック・ハンドラを含めることも可能である。

【００３０】

ソース文字列をターゲット文字列に変換する方法としては、本発明の第２側面による本発明の実施例は、第１文字コード化をもつソース文字列を受信し（ソース文字列は複数のソース文字を含んでいる）、ソース文字列のソース文字の方向を判断し、ソース文字の各々ごとに第２文字コード化に関連する変換コードを、第１文字コード化と判断された方向に基づいてマッピング・テーブルで調べ、ソース文字の変換コードを結合して第２文字コード化のターゲット文字列を形成するオペレーションを実行する。

【００３１】

ソース文字列をターゲット文字列に変換するためのプログラム命令を収めておくコンピュータ可読メディアとしては、本発明の第２側面による本発明の実施例は、第１文字コード化をもつソース文字列を受信することをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列内のソース文字の各々の方向を判断し、ソース文字列をテキスト要素（各テキスト要素はソース文字列の１つまたは複数の文字を含んでいる）に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード

、テキスト要素の各々ごとに第２文字コード化に関連する変換コードを調べることをコンピュータに実行させるように構成されたコンピュータ可読コード、およびテキスト要素の変換コードを結合して第２文字コード化のターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

【００３２】

本発明の第３側面では、コード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するときコンテキストを考慮に入れる。

【００３３】

本発明の第３側面によるコード変換システムは単一ターゲット文字またはターゲット文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかに変換する能力を備えている。変換される文字のコンテキストを判断することにより、コード変換システムは判断された文字のコンテキストを利用してターゲット文字コード化への正しいマッピングが利用されることを保証する。従って、本発明によれば、文字のコンテキストが異なるターゲット・コード化に導くときでも正しいコード変換が達成される。本発明は様々な態様で実現することが可能である。方法、装置またはシステムとして実現することも、コンピュータ可読メディア上に実現することも可能である。

【００３４】

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第３側面による本発明の実施例は、ソース文字コード化をもつ入力文字列（入力文字列は複数の文字を含んでいる）をターゲット文字コード化をもつターゲット文字列に変換することを制御するコンバータ、入力文字列内の文字の各々のコンテキストを判断するスキャナ、ソース・コード化の文字に対するターゲット・コード化をストアしておくマッピング・テーブル、および入力文字列内の文字の各々ごとにターゲット文字コード化に関連する変換コードを、コンテキストと入力文字列内の文字のソース・コード化に基づいて前記マッピング・テーブルで調べるルックアップ・ハンドラを含んでいる。好ましくは、スキャナはさらに入力文字列をテキスト要素（各テキスト要素は入力文字列の１つまたは複数の文字を含んでいる）に分割する。この実施例には、ルックアップ・ハンドラが１つまたは複数のテキスト要素の変換コードを提供できないような、特定の場合にはフォールバック変換コードを提供するフォールバック・ハンドラを含めることも可能である。

【００３５】

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第３側面による本発明の実施例は、ソース文字コード化をもつソース文字列をターゲット文字コード化をもつターゲット文字列に変換することを制御するコンバータ手段、ソース内の文字の各々のコンテキストを判断するステート・マシン、ソース文字コード化の文字に対するターゲット文字コード化をストアしておくマッピング手段、およびソース文字列内の文字の各々ごとにターゲット文字コード化に関連する変換コードを前記マッピング手段で調べるルックアップ・ハンドラ手段を含んでいる。

【００３６】

ソース文字列をターゲット文字列に変換する方法としては、本発明の第３側面による本発明の実施例は、第１文字コード化をもつソース文字列を受信し（ソース文字列は複数のソース文字を含んでいる）、ソース文字列のソース文字の各々のコンテキストを判断し、ソース文字の各々ごとに第２文字コード化に関連する変換コードを、第１文字コード化とソース文字について判断されたコンテキストに基づいてマッピング・テーブルで調べ、ソース文字の変換コードを結合して第２文字コード化のターゲット文字列を形成するオペレーションを実行する。

【００３７】

ソース文字列をターゲット文字列に変換するためのプログラム命令を収めておくコンピュータ可読メディアとしては、本発明の第３側面による本発明の実施例は、第１文字コード化をもつソース文字列を受信することをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列内のソース文字の各々のコンテキストを判断し、ソ

ース文字列をテキスト要素（各テキスト要素はソース文字列の１つまたは複数の文字を含んでいる）に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード、テキスト要素の各々ごとに第２文字コード化に関連する変換コードを調べることをコンピュータに実行させるように構成されたコンピュータ可読コードおよびテキスト要素の変換コードを結合して第２文字コード化のターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

【００３８】

本発明の第４側面では、切り捨て処理手法は、変換のために受信されたソース文字列が、そのソース文字列が変換のためにソース文字列を格納している入力バッファの長さを越えたときでも、異なるターゲット文字コード化に正確に変換されることを保証する。

【００３９】

本発明の第４側面による切り捨て処理は、入力バッファに置かれているソース文字列の一部を切り捨て、その切り捨てられた部分がソース文字列内の後続文字に影響されることなくターゲット文字コード化に変換されるように作用する。当然のことながら、入力バッファに置かれていて、切り捨てが行われた後のソース文字列の部分（切り捨てられた部分）は最大限にしておくと、コード変換が効率よく行えるようになる。本発明は、入力ソース文字列がネットワーク経由で受信されるデータであるときに特に便利である。例えば、データはインターネット経由で電子的に転送されるテキスト・データにすることができる。いずれの場合も、本発明は種々の態様で実現することが可能である。方法、装置またはシステムとして実現することも、コンピュータ可読メディア上に実現することも可能である。

【００４０】

ソース文字列をターゲット文字列に変換するコード変換システムとしては、本発明の第４側面による本発明の実施例は、第１文字コード化をもつソース文字列を第２文字コード化をもつターゲット文字列に変換することを制御するコンバータ、ソース文字列の一部を一度に受け入れるバッファ（ソース文字列は１つまたは複数の文字を含んでいる）、ソース文字列の一部を切り捨てるトランケータ、ソース文字列の切り捨てられた部分をテキスト要素（各テキスト要素はソース文字列の切り捨てられた部分の１つまたは複数の文字を含んでいる）に分割するスキャナ、ソース・コード化のテキスト要素に対するターゲット・コード化をストアしておくマッピング・テーブル、およびテキスト要素の各々ごとに第２文字コード化に関連する変換コードをマッピング・テーブルで調べるルックアップ・ハンドラを含んでいる。

【００４１】

ソース文字列を切り捨ててターゲット文字列に文字変換するための第１方法としては、本発明の第４側面による本発明の実施例は、ソース文字列のバッファ部分をバッファに受け入れ（ソース文字列は２つ以上のバッファ部分を含んでいる）、ソース文字列の後続バッファ部分に影響されることなくターゲット文字列に変換することができる、ソース文字列のバッファ部分のサブパートを判断し、ソース文字列のバッファ部分のサブパートをターゲット・コード化に変換するオペレーションを実行する。好ましくは、第１方法は、変換対象となるバッファ部分の残余部分がサブパートと一緒になったときバッファ部分に等しい場合には、その残余部分を次のバッファ部分またはそのサブパートと一緒に格納するオペレーションも実行する。

【００４２】

ソース文字を切り捨ててターゲット文字列に文字変換するための第２方法としては、本発明の第４側面による本発明の実施例は、ソース文字列の一部をバッファに受け入れ、ソース文字列のその部分内のテキスト要素（各テキスト要素はソース文字列の１つまたは複数の文字を含んでいる）を判断し、テキスト要素が完成しているかどうかを判断し、テキスト要素が完成しているときソース文字列の切り捨てられた部分にテキスト要素を挿入し、ソース文字列の部分が完全に考慮されるまでテキスト要素の判断を繰り返し、そのあと文字変換のためにソース文字列の切り捨てられた部分を出力するオペレーションを実行す

る。好ましくは、この第２方法は、バッファに受け入れられたソース文字列の次の部分と一緒に使用するために、ソース文字列の残余部分を格納しておくオペレーションも実行する。

【００４３】

ソース文字列をターゲット文字列に変換するためのプログラム命令を格納しておくコンピュータ可読メディアとしては、本発明の第４側面による本発明の実施例は、第１文字コード化をもつソース文字列の一部をバッファに受け入れることをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列の一部を切り捨てることをコンピュータに実行させるように構成されたコンピュータ可読コード、ソース文字列の切り捨てられた部分をテキスト要素（各テキスト要素はソース文字列の切り捨てられた部分の１つまたは複数の文字を含んでいる）に分割することをコンピュータに実行させるように構成されたコンピュータ可読コード、テキスト要素の各々ごとに第２文字コード化に関連する変換コードを調べることをコンピュータに実行させるように構成されたコンピュータ可読コード、およびテキスト要素の変換コードを結合して第２文字コード化のターゲット文字列を形成することをコンピュータに実行させるように構成されたコンピュータ可読コードを含んでいる。

【００４４】

本発明のその他の側面および利点は、本発明の原理を例示している添付図面を参照して以下に詳述している説明の中で明らかにする。

【００４５】

本発明の理解を容易にするために、以下では、添付図面を参照して本発明を詳しく説明する。なお、図面において、類似する構成要素は類似の参照符号を付けて示されている。

【発明を実施するための最良の形態】

【００４６】

以下、図１ないし図２９を参照して本発明の実施例について説明する。なお、この分野の精通者ならば理解されるように、これらの図を参照して以下に詳述する説明は本発明の理解を容易にすることを目的とし、本発明は以下に説明する実施例に限定されるものではない。

【００４７】

本発明の第１側面によるコード変換システムはソース文字を異なるコード化のターゲット文字に変換するものである。本発明によるコード変換システムはラウンドトリップ忠実度を備えていると共に、その結果の文字コードが他のプラットフォームと互換性をもつことを保証する。コード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする機能を備えている。ラウンドトリップ忠実度により、ソース・テキストはターゲット・テキストに変換し、そのあとオリジナル・ソース・テキストに戻すように再変換することができる。その結果の文字コードが他のプラットフォームと互換性をもつことは、標準ターゲット文字の使用を最大限にし、プライベート文字の使用を最小限にすることによって保証される。コード変換システムは他の文字セットからユニコード文字に変換し、ユニコード文字を他の文字セットに変換するときに利用すると、特に便利である。ユニコード文字シーケンスをターゲット文字セット内の単一文字にマッピングすることは従来にはなかったものである。

【００４８】

コード変換システムはコンピュータ・システムや他のエレクトロニック・デバイスにしてこれらのコード変換オペレーションを行うことができる。このコンピュータ・システムは必要とする目的だけに使用するように構築することも、コンピュータ・プログラムに従って動作する汎用コンピュータにすることも可能である。以下に説明する処理はどのコンピュータ・システムや他のエレクトロニック・デバイスにも適用可能である。具体的には、種々の汎用コンピューティング・マシンは以下に教示する説明に従って書かれたソフトウェアと共に使用することができるが、もっと特殊化されたエレクトロニック・デバイスを構築すると、必要とするオペレーションを実行できるので好都合である。

【００４９】
本発明の第２側面によるコード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するとき方向を考慮に入れる。このコード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスにマッピングする機能を備えている。変換される文字の方向を判断または解明することにより、コード変換システムは判断または解明された文字の方向を利用してターゲット文字コード化への正しいマッピングが得られることを保証する。従って、本発明によれば、ソース文字列内の文字の方向が変化するときでも正しいコード変換が達成される。
【００５０】
本発明がアラビアまたはヘブライ・ベースの文字セットが使用されるときに特に便利であるのは、これらの文字が右から左への方向をもっているからである。本発明によれば、どちらかの方向を取り扱うときの柔軟性が得られるだけでなく方向を途中で変更できるという機能も得られる。方向を途中で変更できることはアラビア文字および／またはヘブライ文字が左から右への方向が最も普通である他の文字セットと一緒に使用されるような場合に重要である。異なる方向をもつ文字の使用例として、スペース文字がある。ユニコードでは、コード化は１つだけであり、固有の方向をもっていない。他方、ＭａｃＡｒａｂｉｃでは、左から右へのスペース文字と右から左へのスペース文字の２種類がある。
【００５１】
変換されるソース文字の方向を判断または解明することは、方向解明手法によって達成されるが、この手法についてはソース文字を異なるコード化のターゲット文字に変換するコード変換システムと関連づけて以下で詳しく説明する。
【００５２】
本発明の第３側面によるコード変換システムはソース文字コード化からの文字をターゲット文字コード化に変換するときコンテキストを考慮に入れる。このコード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする機能を備えている。文字のコンテキストを判断することにより、コード変換システムは判断された文字のコンテキストを利用してターゲット・コード化への正しいマッピングが得られることを保証する。従って、本発明によれば、文字のコンテキストがターゲット・コード化へのマッピングに影響するときでも正しいコード変換が達成される。
【００５３】
ある種の文字セットでは、文字が使用されているコンテキストによって文字が異なる表現形態をもつことになる。これらの文字の表現形態は絵文字(glyphs)である。異なる表現形態は表示されるとき異なって現れる。文字が使用されているコンテキストによって表現形態が決まる。アラビア文字はそのコンテキストに応じて異なる表現形態をもつ言語である。
【００５４】
アラビア・スクリプトはアラビア語を書くために使用される。アラビア・スクリプトはその印刷形体においても筆写体（カーシブ）である。その結果として、同じ文字はそれが隣の文字とどのように結合されるかに応じて異なった形体で書かれることがある。このような文字の例として、アラビア文字“ＨＥＨ”（ユニコード文字ではｕ０６４７とコード化される）がある。例えば、ユニコードでのアラビア文字はＤＯＳアラビア文字の４つの異なる表現形態の１つをコンテキストに応じてマッピングされる場合がある。その結果、本発明によるコード変換システムはユニコード・ストリーム内のアラビア文字のコンテキストを判断し、正しいマッピングが得られるように動作する。
【００５５】
変換されるソース文字のコンテキストの判断はコンテキスト処理によって達成されるが、このコンテキスト処理はソース文字を異なるコード化のターゲット文字に変換するコード変換システムと関連づけて以下で詳しく説明する。
【００５６】

本発明の第２および第３側面によれば、コード変換システムはソース文字を異なるコード化のターゲット文字に変換する。本発明によるコード変換システムはラウンドトリップ忠実度を提供すると共に、その結果の文字コードが他のプラットフォームと互換性をもつことを保証する。コード変換システムは単一ソース文字または文字シーケンスを単一ターゲット文字またはターゲット文字シーケンスのどちらかにマッピングする機能を備えている。ラウンドトリップ忠実度によると、ソース・テキストはターゲット・テキストに変換し、そのあとオリジナル・ソース・テキストに戻るように再変換することができる。その結果の文字コードと他のプラットフォームとの互換性は標準ターゲット文字の使用を最大限にし、プライベート文字の使用を最小限にすることによって保証される。コード変換システムは他の文字セットからユニコード文字に変換し、ユニコード文字を他の文字セットに変換するときに利用すると、特に便利である。ユニコード文字シーケンスをターゲット文字セット内の単一文字にマッピングすることは従来にはなかったものである。

【００５７】

コード変換システムはコンピュータ・システムや他のエレクトロニック・デバイスにしてこれらのコード変換オペレーションを行うことができる。このコンピュータ・システムは必要とする目的だけに使用するように構築することも、コンピュータ・プログラムに従って動作する汎用コンピュータにすることも可能である。以下に説明する処理はどのコンピュータ・システムや他のエレクトロニック・デバイスにも適用可能である。具体的には、種々の汎用コンピューティング・マシンは以下に教示する説明に従って書かれたソフトウェアと共に使用することができるが、もっと特殊化されたエレクトロニック・デバイスを構築すると、必要とするオペレーションを実行できるので好都合である。

【００５８】

本発明の第４側面では、切り捨て処理手法は変換のために受信されたソース文字列が、そのソース文字列が変換のためにソース文字列を格納している入力バッファの長さを越えたときでも、異なるターゲット文字コード化に正確に変換されることを保証する。切り捨て処理手法は入力バッファに置かれているソース文字列の一部を切り捨てて、切り捨てられた部分がソース文字列内の後続文字に影響されることなくターゲット文字コード化に変換されるように作用する。当然のことであるが、入力バッファに置かれていて、切り捨てられた後のソース文字列の部分（切り捨てられた部分）を最大限にすると、コード変換を効率よく実行することができる。

【００５９】

本発明による切り捨て処理手法はソース文字を異なるコード化のターゲット文字に変換するコード変換システムと関連づけて以下で詳しく説明する。

【００６０】

図１は本発明による代表的なコンピュータ・システム１００を示すブロック図である。コンピュータ・システム１００は中央処理ユニット（ＣＰＵ）１０２を装備し、このＣＰＵは双方向にランダムアクセス・メモリ（ＲＡＭ）１０２と、単方向にリードオンリ・メモリ（ＲＯＭ）１０６と結合されている。代表例として、ＲＡＭ１０４はプログラミング命令とデータを格納し、この中には現在ＣＰＵ１０２上で実行中のプロセスに対する他のデータや命令のほかに、以下で説明するテーブルが含まれている。代表例として、ＲＯＭ１０６はコンピュータ・システムがその機能を実行するために使用する基本的操作命令、データおよびオブジェクトを格納している。そのほかに、ハードディスク、ＣＤ　ＲＯＭ、磁気光学（フロプチカル）ドライブ、テープ・ドライブなどの大量記憶装置１０８は双方向にＣＰＵ１０２と結合されている。大量記憶装置１０８は一般的にＣＰＵによって活発に使用されることがない追加のプログラミング命令、データおよびテキスト・オブジェクトを収容しているが、アドレス空間は例えば仮想メモリなどのためにＣＰＵによってアクセス可能である。上述したコンピュータの各々はさらに入出力ソース１１０を含んでおり、その代表例として、キーボードやポインタ・デバイス（例えば、マウスやスタイラス）などの入力メディアがある。コンピュータ・システム１００はデータと命令を転送できるネットワーク接続１１２を含むこともできる。追加の大量記憶装置（図示せず）をネッ

トワーク接続１１２を経由してＣＰＵ１０２に接続することも可能である。コンピュータ・システム１００はさらに、コンピュータ・システム１００によって生成または表示されたテキストおよびイメージ（画像など）を見るための表示スクリーン１１４を含んでいる。

【００６１】

ＣＰＵ１０２はオペレーティング・システム（図示せず）と一緒になって、コンピュータ・コードを実行するように動作する。コンピュータ・コードはＲＡＭ１０４、ＲＯＭ１０６、または大量記憶装置１０８に置いておくことができる。また、コンピュータ・コードはポータブル・プログラム・メディア１１６に置いておき、必要時にコンピュータ・システム１００にロードまたはインストールすることも可能である。ポータブル・プログラム・メディア１１６の例としては、ＣＤ－ＲＯＭ、ＰＣカード・デバイス、ＲＡＭデバイス、フロッピディスク、磁気テープなどがある。

【００６２】

Ｉ．定義

１．コード・ポイント：コード・ポイントとは、特定のコード化におけるビット・パターンである。通常、ビット・パターンは１または２バイト以上の長さになっている。ユニコードのコード・ポイントは常に１６ビットまたは２バイトである。

【００６３】

２．コード化：コード化とは、文字セットとコード・ポイントのセットとを１対１で対応づけたもの（マッピング）である。例えば、ＡＳＣＩＩコード化はａ－ｚ、Ａ－Ｚ、および０－９を含むセットをコード・ポイントｘ００－ｘ７Ｆに対応づけている。

【００６４】

３．テキスト要素：テキスト要素は特定のオペレーションで１つの単位として扱われる１つまたは２つ以上のコード・ポイントのシーケンスである。例えば、ＬＡＴＩＮ　ＣＡＰＩＴＡＬ　ＬＥＴＴＥＲ　Ｕとその後に続くＮＯＮ－ＳＰＡＣＩＮＧ　ＤＩＡＥＲＥＳＩＳは本発明によるコード変換オペレーションの対象となるテキスト要素である（例えば、この例では、２つの隣り合う文字）。

【００６５】

４．絵文字：文字を可視的に表現した表示形態。例えば、イタリックの“*a*”とローマン字の“ａ”は同じ基礎文字を表現する２つの異なる絵文字である。

【００６６】

５．表現形態：表現形態はコンテキストに応じてその可視形体を変化させる絵文字である。ある種のコード化はコンテキストから独立している抽象文字だけをマッピングするのに対し、他のコード化は表現形体だけをマッピングする。例えば、“ｆｉ”のような合字は、ＬＡＴＩＮ　ＣＡＰＩＴＡＬ　ＬＥＴＴＥＲ　Ｆとその後に続くＬＡＴＩＮ　ＣＡＰＩＴＡＬ　ＬＥＴＴＥＲ　Ｉの文字シーケンスの表現形態である。

【００６７】

６．フォールバック：フォールバックはソース・コードに正確に等価ではないが、オリジナルの情報の一部を残しているターゲット・コード化内の１つまたは２つ以上のコード・ポイントのシーケンスである。例えば、（Ｃ）は©を表す可能なフォールバックである。

【００６８】

７．デフォルト：デフォルトはターゲット・コード化内にソース・コード・ポイントに類似したものがないとき使用されるターゲット・コード化内の１つまたは２つ以上のコード・ポイントのシーケンスである。

【００６９】
II. ユニコード・コンバータ
本発明による一般的変換手法はソース文字を異なるコード化のターゲット文字に変換する。好ましくは、ソース文字またはターゲット文字のどちらかはユニコード文字になっている。
【００７０】
ユニコード標準は開発されたいくつかの文字コード化を汎用国際的文字コード化標準に統一化したものである。図２はユニコード文字コード化のフォーマットを示す図である。具体的には、ユニコード標準に用意されているコードは、図２に示すフォーマット２００で示すように１６ビット幅になっている。本明細書では、ユニコード文字は頭にｕを付けた１６進数で表され（例えば、ｕ００１）、他のコード化内の文字は頭にｘを付けた１６進数で表されている（例えば、ｘ４１は１バイト文字を表し、ｘ８１４０は２バイト文字を表している）。
【００７１】
図３は、ソース文字列３０２を受信し、ターゲット文字列３０４を出力する本発明による基本的ユニコード・コード変換システム３００を示すブロック図である。ユニコード・コード変換システム３００はソース文字列３０２の文字を、ターゲット・ストリーム内にあって、文字コード化がソース文字列で使用されたコード化と異なっている１つまたは複数の文字に変換するように動作する。好ましくは、ユニコード・コード変換システム３００はユニコードから異なるターゲット・コード化に変換するか（ユニコードから）、あるいは異なるソース・コードから変換する（ユニコードへ）。
【００７２】
図４は本発明によるユニコード・コード変換システム４００の実施例を示すブロック図である。本発明の第４側面では、トランケータ４０７とバッファ４０５が存在し、適用されているが、第１、第２または第３側面では、これらは必ずしも存在する必要がないので、必ずしも適用されるとは限らない。
【００７３】
本発明の第１、第２および第３側面によれば、ユニコード・コード変換システム４００はユニコード文字列４０４を受信し、ターゲット文字列４０６を出力するユニコードからのコンバータ４０２を含んでいる。ユニコードからのコンバータ４０２は本発明に従ってコード変換プロセスを実行する。その際に、ユニコードからのコンバータ４０２はスキャナ４０８とやりとりする。スキャナ４０８はスキャナ・テーブル４０８と一緒に使用されて、ユニコード文字列をスキャンしテキスト要素を特定する。そのあと、ユニコードからのコンバータ４０２はルックアップ・ハンドラ４１２を使用して、スキャナ４０８によって特定されたテキスト要素のターゲット・コード化内の１つまたは複数の文字を調べる。ルックアップ・ハンドラ４１２はマッピング・テーブル４１４を使用してテキスト要素のターゲット・コード化内の１つまたは複数の文字を取得する。さらに、ユニコードからのコンバータ４０２はフォールバック・ハンドラ４１６を使用することも可能である。フォールバック・ハンドラ４１６はマッピング・テーブル４１４と一緒に使用され、ルックアップ・ハンドラ４１２がテキスト要素のターゲット・コード化内の１つまたは複数の文字を特定できなかった場合に、ターゲット・コード化内にあって、テキスト要素のフォールバック・マッピングとして使用できる１つまたは複数の文字を特定するように動作する。状態管理機構(state administrator)４１８は変換の現在状態に関する情報を維持し、あるいはストアしている。この情報の例としては、シメトリック・スワッピングのコンテキスト、方向および状態がある。
【００７４】
本発明の第４側面によれば、ユニコード・コード変換システム４００はユニコード文字列４０４を受信し、ターゲット文字列４０６を出力するユニコードからのコンバータ４０２を含んでいる。ユニコード文字列はバッファ４０５にストアされる。トランケータ４０７はバッファ４０５にストアされたユニコード文字列４０４の長さを切り捨てて、正確な

変換が行われることを保証する。

【００７５】

ユニコードからのコンバータ４０２はコード変換の全体的オペレーションを制御する。その際に、ユニコードからのコンバータ４０２はスキャナ４０８とやりとりする。スキャナ４０８はスキャナ・テーブル４０８と一緒に使用されて、切り捨てられたユニコード文字列４０４（トランケータ４０７から与えられたもの）をスキャンしテキスト要素を特定する。そのあと、ユニコードからのコンバータ４０２はルックアップ・ハンドラ４１２を使用して、スキャナ４０８によって特定されたテキスト要素のターゲット・コード化内の１つまたは複数の文字を調べる。ルックアップ・ハンドラ４１２はマッピング・テーブル４１４を使用してテキスト要素のターゲット・コード化内の１つまたは複数の文字を取得する。さらに、ユニコードからのコンバータ４０２はフォールバック・ハンドラ４１６を使用することも可能である。フォールバック・ハンドラ４１６はマッピング・テーブル４１４と一緒に使用され、ルックアップ・ハンドラ４１２がテキスト要素のターゲット・コード化内の１つまたは複数の文字を特定できなかった場合に、ターゲット・コード化内にあって、テキスト要素のフォールバック・マッピングとして使用できる１つまたは複数の文字を特定するように動作する。状態管理機構(state administrator)４１８は変換の現在状態に関する情報を維持し、あるいはストアしている。この情報の例としては、シメトリック・スワッピングのコンテキスト、方向および状態がある。この種の情報は、切り捨てられたユニコード文字列がブロックの終わりで終わっていない非ブロック区切り変換を行うとき必要になるものである。そのような場合には、変換の現在状態に関する情報をストアしておくと、コード変換プロセスは入力文字列４０４がバッファ４０５のサイズよりも大きいときでも、正確なコード変換を行うことができる。

【００７６】

Ａ．スキャナおよびスキャナ・テーブル

スキャナ４０８はスキャナ・テーブル４１０と一緒に使用されて、ユニコード文字列４０４をスキャンし、ルックアップ・ハンドラ４１２が必要とする次のテキスト要素と追加情報を戻す。追加情報には、方向情報、コンテキスト情報、および種々の状態インジケータの１つまたは２つ以上が含まれている。以下では、スキャナ４０８の一般的オペレーションについて説明する。スキャナ４０８は入力ユニコード文字列４０４の文字をスキャンして行く。ターゲット・コード化のために方向情報が必要であれば、テキスト要素内の各文字ごとに文字方向が取得される。また、ターゲット・コード化のために文字コンテキスト情報が必要であれば、テキスト要素内の各文字ごとに文字コンテキスト情報が取得される。そのあと、スキャナ４０８が文字の各々をスキャンしていくとき、スキャナ４０８はスキャナ・テーブル４１０に入っている情報に従って文字に対するアクションをとる。スキャナ４０８がどのようなアクションをとるかは、状態と文字クラスに基づいて判断される。スキャナ４０８がとることができるアクションとしては、現在の文字にマークを付けること、シメトリック・スワッピング・ビットをセットまたはクリアすること、テキスト要素のコンテキスト形式を記録すること、テキスト要素が再配列を必要とすることを示すフラグをセットすること、テキスト要素の終わりを示すこと、などがある。シメトリック・スワッピング・ビット、コンテキストおよび方向はスキャナの状態に関する情報として状態管理機構４１８によってセーブされる。戻す前に、スキャナ４０８はテキスト要素のコンテキスト情報をセーブしておく。スキャナ４０８はテキスト要素（入力文字列内の各テキスト要素）とその属性を返却する。属性には次のものがある。方向、クラス、優先順位、シメトリック・スワッピング状態、サブセットおよびコンテキストである。スキャナ４０８がテキスト要素を判断したあと、文字を標準形順序に再配列する必要が起こる場合がある。１つの例として、テキスト要素内の文字の再配列はユニコードで定義されている標準形順序になっていない非スペース・マークがテキスト要素に含まれているときに行われる。

【００７７】

好ましくは、スキャナ４０８はスキャナ・テーブル４１０と一緒に、並列に動作するペ

アのステート・マシンとして実現されている。第１ステート・マシンは文字方向を解明し、第２ステート・マシンは該当する場合には、テキスト要素と文字形式コンテキスト情報を計算し、シメトリック・スワッピング状態も記録しておく。別々になった２つのステート・マシンを使用すると、ユニコード・コード変換システム４００の設計と保守が容易化される。第１および第２ステート・マシンは状態とクラスによってインデックスされる２次元配列（またはテーブル）として実現することができる。スキャナ４０８がとるべきアクションが文字方向によって決まる場合には、ステート・マシン・エントリは各方向についてスキャナ４０８がとるべき該当アクションを収めている別のテーブルまでのインデックスである。

【００７８】

スキャナ４０８の機能は入力ユニコード文字列４０４をテキスト要素に変換し、テキスト要素とその属性を戻すことである。スキャナ４０８はテキスト要素のある種の特性をセーブしておく必要があり、そうすれば、ターゲット・コード化で正しく変換されることになる。すなわち、特性には、方向、コンテキストおよびシメトリック・スワッピング状態がある。しかし、スキャナ４０８は、そのオペレーションが特定のターゲット・コード化から独立しているので、どのようなターゲット・コード化であるかを知っている必要はない。それにもかかわらず、ユニコード変換システム４００は、テキスト要素の定義（つまり、チャンク行動）が、スキャナ・テーブル４１０を変更するだけでターゲット・コード化と共に変化できるように実現されていることが好ましい。

【００７９】

文字の方向性は文字を表示するために使用される。例えば、アラビアまたはヘブライ文字が表示スクリーンに表示されるとき、これらは右から左への順序になっている。大部分のユニコード文字は黙示的方向をもっている（ユニコード・バージョン１．０の４０７ページ（セクション４．６）および６１１ページ（付録Ａ）を参照）。ユニコード標準に用意されている暗黙的方向クラスとその値には次のものがある。左から右へ（０）、右から左へ（１）、ヨーロッパ数字（２）、ヨーロッパ数字セパレータ（３）、ヨーロッパ数字ターミネータ（４）、アラビア数字（５）、共通数字セパレータ（６）、ブロック・セパレータ（７）、セグメント・セパレータ（８）、ホワイトスペース（９）およびその他の数字である。スキャナ４０８はテキスト要素の文字の方向クラスを調べる。次に、その方向クラスはテキスト要素の方向を解明するために使用される。また、方向性をオーバライドまたは埋め込ませる特殊なユニコード文字もある。これらの特殊なユニコード文字はスキャナ４０８によって単一文字テキスト要素として扱われる。

【００８０】

テキスト要素を形成するときスキャナ４０８が従う基本的ルールがいくつかある。ベース・ルールでは、適用されるルールがなければ、テキスト要素は単一ユニコード文字とされる。別のルールでは、ベース文字に続く非スペースまたは結合マークはそのベース文字と一緒に単一テキスト要素として分類される。さらに別のルールでは、シンボル（つまり、朝鮮ハングル・ジャモス文字）、合字または表意文字に関連する文字が見つかると、これらはテキスト要素に結合される。さらに別のルールでは、フラクション・スラッシュの両側が１つまたは複数の１０進数のシーケンスで囲まれていると、これらは数字フラクション・テキスト要素として結合される。

【００８１】

以下では、非スペースまたは結合マークに関するルールについて詳しく説明する。ユニコード標準によれば、非スペース・マークはベース文字のあとに置かれている。従って、ベース文字のあとに置かれた非スペース文字はベース文字を含むテキスト要素の一部となる（ユニコード標準バージョン１．０、４０３ページ（セクション４．３）を参照）。例えば、単一の非スペース文字の後に非スペース文字でない文字が続いているときは、その非スペース文字は直前の文字と一緒にテキスト要素として結合される。その場合、テキスト要素の長さは２であり、テキスト要素の属性はベース文字によって定義されている。前に置かれた文字がなければ、非スペース文字は単一テキスト要素として渡されるだけであ

る。複数の非スペース文字もこのようにして結合することができる。

【００８２】

以下では、朝鮮ハングル・ジャモス文字について詳しく説明する。各ハングル文字は暗黙値を持ち、これは次のクラスの１つになっている。Ｃｈｏｓｅｏｎｇ（初期）、Ｊｕｎｇｓｅｏｎｇ（中間）またはＪｏｎｇｓｅｏｎｇ（最終）である。ユニコード標準バージョン１．１（セクション５）はこれらの文字のコードと許容される組み合わせをリストしている。スキャナ４０８はこれらの文字の許容される組み合わせに従って朝鮮ジャモス文字をグループ化する。入力が許容されない組み合わせのときは、スキャナ４０８は文字を単一テキスト要素として返却する。前述したように、ハングル分節のあとに結合マークが付いているときは、その結合マークはハングル分節のテキスト要素内に挿入される。

【００８３】

次に、数字フラクションに関するルールについて詳しく説明する。スキャナ４０８は最初にフラクション・スラッシュ数字の各文字を、それらが単一文字テキスト要素であるものとして取り扱う。しかし、完全なフラクション・スラッシュ・シーケンスが見つかったときは、スキャナ４０８はそのシーケンスに関連する文字を単一テキスト要素になるように結合する。ディジットが結合マークと一緒に見つかったときは、そのディジットと結合マークはフラクション・スラッシュの一部となることができないが、ディジットと結合マークは一緒にテキスト要素を形成することができる。

【００８４】

非スペース文字を除き、すべてのアラビア文字は単一テキスト要素としてスキャナ４０８を通過する。アラビア形式のシェーピング状態文字も、単一テキスト要素としてスキャナを通過する。方向フォーマッティング・コードは単一テキスト要素としてスキャナ４０８を通過する。

【００８５】

Ｂ．ルックアップ・ハンドラ、マッピング・テーブルおよびフォールバック・ハンドラ

マッピング・テーブル４１４は１つまたは複数のユニコード文字の入力シーケンスをターゲット・コード化における１つまたは複数の出力シーケンスと突き合わせるためにルックアップ・ハンドラ４１２によって使用される。ユニコード・シーケンス（つまり、テキスト要素）自体のほかに、入力シーケンスに関するある種の追加情報が得られ（例えば、方向、コンテキスト、シメトリック・スワッピング状態、垂直形式要求、フォールバック要求、許容範囲、変種）、ある種のテーブルはこの情報を利用している。好ましくは、マッピング・テーブル４１４はフォールバック・ハンドラ４１６が必要とするデータもストアしているが、別のテーブルを用意してフォールバック・ハンドラ４１６に使用させることも可能である。

【００８６】

図５はユニコード・コード変換システム４００のマッピング・テーブル４１４の好ましい配列を示す概略図である。マッピング・テーブル４１４は好ましくはヘッダ部分５００を含み、マッピング・テーブル４１４のデータのセグメントはテキスト要素内の文字数に基づいて分割されている。ヘッダ部分５００の内容は以下で詳しく説明する。図５に示すマッピング・テーブル４１４は１からＮ文字までのテキスト要素のコード化をサポートしている。ルックアップ・ハンドラ４１２が１文字テキスト要素のターゲット・コード化を探すためにマッピング・テーブル４１４をサーチするとき、マッピング・テーブル４１４のセグメント５０２が使用される。同様に、テキスト要素が２文字幅であれば、セグメント５０４が使用され、テキスト要素がＮ文字幅であれば、セグメント５０６が使用される。図４と図５は単一マッピング・テーブル４１４を示しているが、ユニコード・コード変換システム４００は複数の異なるマッピング・テーブル４１４を使用する。つまり、各ターゲット文字セットごとに１つのマッピング・テーブルを使用する。各マッピング・テーブルは複数のサブテーブルを含んでいる。

【００８７】

マッピング・テーブル４１４はサイズと全体的変換速度要件を考慮に入れて設計されて

いる。マッピング・テーブル４１４はルックアップ時間を重大に低下させることなく可能な限り小さくしておくべきであり、ルックアップ時間はテーブル・サイズを大幅に大きくすることなく可能な限り高速にしておくべきである。ユニコード・コード変換システム４００は複数のテーブル・フォーマットをサポートしているので、各サブテーブルごとに異なるフォーマットにすることが可能である。そのようにすると、速度とサイズのトレードオフを特定のテーブルに合わせて必要時に調整することができる。好ましくは、マッピング・テーブル４１４の設計は単一ユニコード文字からターゲット・コード化内の単一文字にマッピングするのを可能な限り高速化するものでなければならないが、これは最も普通の使い方であるからである。

【００８８】

マッピング・テーブル４１４の設計は、テーブルがフォールバック・ハンドラ４１６の要求の少なくとも一部をサポートし、複数のマッピング許容範囲をサポートし、複数のターゲット文字セットの変種をサポートするようになっている。テーブル・フォーマットは１つまたは複数のユニコード文字シーケンスをゼロまたは１個以上の文字の出力シーケンスにマッピングすることもできる。マッピング・テーブル４１４は単一入力シーケンスに対して起こり得る複数の出力シーケンスを指定することが可能であり、特定の出力シーケンスは方向、コンテキストおよびシメトリック・スワッピング状態などの属性によって判断される。本発明の第３側面に関しては、中心となるのは出力シーケンスの１つをコンテキストに基づいて選択することである。マッピング・テーブルは容易に拡張することができるので、ユニコード・コード変換システム４００のコード化の振舞を容易にカストマイズすることができる。

【００８９】

マッピング・テーブル４１４が必要とする情報には次のものがある。スキャナ４０８からのテキスト要素（つまり、結合マークを標準形順序にして変換される入力文字シーケンス）、垂直形式が使用可能なときに水平形式の代わりに使用するかどうか、解明されたテキスト要素の方向、テキスト要素の文字形式コンテキスト情報（初期、中間、最終、または隔離）、シメトリック・スワッピングの現在状態、どのレベルのルックアップを起動させるかの情報（許容レベル（厳格または緩和）とフォールバック（オンまたはオフ））、および特定のコード化変種の識別子である。ルックアップ・レベルに関する情報はコールまたはユニコード・コード変換システム４００をコールするアプリケーション・プログラムから与えられる。方向およびコンテキストが重要でない言語または文字では、解明された方向とコンテキスト情報は必要でない。

【００９０】

変種の定義、ユニコード・シーケンスからターゲット・シーケンスへの実際のマッピング、およびこれらにアクセスするために使用されるテーブル・フォーマットはマッピング・テーブル４１４の設計によって変更可能である。従って、正確性および若干であるが、パフォーマンスとサイズとのトレードオフはマッピング・テーブル４１４の設計に大きく依存している。好ましいことは、マッピング・テーブル４１４が厳格および緩和マッピング、フォールバック・マッピング、およびデフォルト・マッピングをサポートすることである。

【００９１】

厳格マッピングはラウンドトリップ忠実度が保証されるコード変換である。ユニコードからターゲット文字セットへの厳格マッピングはその文字セットからユニコードへのマッピングとは逆である。ユニコードからターゲット文字セットへの緩和マッピングはターゲット文字セット内の文字の定義または隔離された用途の範囲に属する追加マッピングである。緩和マッピングは正しくマッピングされるように見えるが、若干のあいまいさがある。例えば、多くの文字セットでは、単一文字は、明示的定義、あいまいな定義、または確立された用法のいずれかによって複数の意味を持つことがある。例えば、Ｓｈｉｆｔ－ＪＩＳ文字ｘ８１６１は２つの意味を持つように規定されている。すなわち、「２重垂直線」と「平行」である。これらの意味の各々は異なるユニコード文字ｕ２０１６「２重垂直

線」とｕ２２２５「に平行」に対応している。Ｓｈｉｆｔ－ＪＩＳからユニコードにマッピングするときは、コード変換システムはこれらのユニコード文字の１つ、つまり、２重垂直線を選択しなければならない。ユニコードからＳｈｉｆｔ－ＪＩＳにマッピングするときは、コード変換システムは両方のユニコード文字を同一Ｓｈｉｆｔ－ＪＩＳ文字にマッピングすることができ、そのようにするのが通常である。しかし、これらのユニコードからのマッピングの１つだけはユニコードへのマッピングとは逆になっている。

【００９２】

厳格マッピングと緩和マッピングの比較例

・　ユニコードｕ０００ＤがＡＳＣＩＩｘ０Ｄ「キャリッジリターン」に厳格にマッピングされていれば、ユニコードｕ２０２９「パラグラフ・セパレータ」はＡＳＣＩＩｘ０Ｄにゆるやかにマッピングすることができる。

・　ユニコードｕ００２Ｄ「ハイフン－マイナス」がＡＳＣＩＩｘ２Ｄ「ハイフン－マイナス」に厳格にマッピングされていれば、ユニコードｕ２０１０「ハイフン」とｕ２２１２「マイナス記号」はＡＳＣＩＩｘ２Ｄにゆるやかにマッピングすることができる。

・　ユニコードｕ００ＥＯ「グラーブ付きのラテン小文字Ａ」がＩＳＯ　８８５９－１ｘＥＯ「アクサングラーブ付きの小文字ａ」に厳格にマッピングされていれば、２文字のユニコード・シーケンスｕ００６１＋ｕ０３００「ラテン小文字Ａ」＋「結合アクサングラーブ」はＩＳＯ　８８５９－１ｘＥＯにゆるやかにマッピングすることができる。

・　Ｓｈｉｆｔ－ＪＩＳは半幅文字と全幅文字を区別しているので、Ｓｈｉｆｔ－ＪＩＳの緩和マッピングもこれらを区別しておかなければならない。つまり、ユニコードｕＦＦ４０「全幅アクサングラーブ」はＳｈｉｆｔ－ＪＩＳｘ８１４Ｄ「アクサングラーブ（全幅）に厳格にマッピングされており、これはＳｈｉｆｔ－ＪＩＳｘ６０「アクサングラーブ（半幅）と区別されている。ユニコード・シーケンスｕ３０００＋ｕ０３００「表意文字スペース」＋「結合アクサングラーブ」はＳｈｉｆｔ－ＪＩＳｘ８１４Ｄにゆるやかにマッピングすることができる。しかし、ユニコード・シーケンスｕ００２０＋ｕ０３００「スペース」＋「結合アクサングラーブ」はＳｈｉｆｔ－ＪＩＳｘ８１４Ｄにゆるやかにマッピングしてはならない。これはＳｈｉｆｔ－ＪＩＳｘ６０にマッピングされるべきである。

【００９３】

本発明の第１側面によれば、ユニコードからある種の他の文字にマッピングし再び元に戻すマッピング（ラウンドトリップ・マッピング）は、他の文字への厳格なマッピングが存在しているユニコード文字だけを使用するとき可能である。

【００９４】

さらに、本発明の第１から第５までの側面によれば、フォールバック・マッピングはユニコード文字の意味または同一性を保存していないユニコードからのマッピングである。つまり、これらはユニコード文字（または文字シーケンス）をその定義または用途がユニコード文字の意味または用途を含んでいないターゲット文字セット内の文字（または文字シーケンス）にマッピングする。これに対して、フォールバック・マッピングによると、ユニコード文字（または文字シーケンス）に最も近く対応しているターゲット・コード化内の文字（または文字シーケンス）が得られる。

【００９５】

フォールバック・マッピングの例

・　ユニコード文字ｕ０３００「結合アクサングラーブ」はフォールバック・マッピングとしてＡＳＣＩＩｘ６０「アクサングラーブ［スペース］）にマッピングすることができる。違いは、ユニコード文字が結合マーク（非スペース）であるのに対し、ＡＳＣＩＩ文字はスペース・マークであることである。

・　ユニコード文字ｕ０１Ｃ０「ラテン文字デンタル・クリック」はフォールバック・マッピングとしてＡＳＣＩＩｘ７Ｃ「垂直線」にマッピングすることが可能である。

・　ユニコード文字ｕ２００１「ＥＭ　ＱＵＡＤ」はフォールバック・マッピングとしてＡＳＣＩＩｘ２０「スペース」にマッピングすることが可能である。

【００９６】
従って、上記の例に示すように、フォールバック・マッピングはユニコード文字（またはシーケンス）にグラフィックが近似しているターゲット文字（またはシーケンス）を生成するために使用される。
【００９７】
パフォーマンス上の理由により（つまり、コード化をマッピング・テーブルから得るときの速度）、マッピング・テーブル４１４までのインデックスにはいくつかのフォーマットがある。可能とされるフォーマットには、セグメント・フォーマット、リスト・フォーマット、範囲フォーマット、チェーン・フォーマットがある。文字シーケンスの長さが異なるごとに、別々のインデックスを設けることが好ましい。その結果として、セグメント５０２、５０４、５０６の各々に関連するインデックスは異なるフォーマットにすることができ、各インデックスの先頭の情報はそのフォーマットを指定している。フォーマットに関係なく、各インデックスは最終的には、入力シーケンスを直接に出力シーケンスにマッピングするか、あるいは出力シーケンスが長ければ、対応する出力シーケンスのロケーションを指定しているオフセットにマッピングする。
【００９８】
マッピング・テーブル４１４までのインデックスのチェーン・フォーマットについては、以下で詳しく説明する。チェーン・フォーマットでは、セクションの先頭がチェックされ、それがチェーン・フォーマット・テーブルのチェーン・ヘッダであるか、他のフォーマットであるかが判断される。チェーン・フォーマットは複数のインデックス・テーブルのチェーンを指定し、各々は異なるフォーマットになっている場合がある。必要とするマッピングが最初のインデックス・テーブルに見つからなければ、ルックアップ・ハンドラ４１２は２番目のインデックス・テーブルを調べ、以下同様である。チェーン・フォーマットは、例えば、あるインデックス・フォーマット（これは空間的および／または時間的に効率的である）が入力シーケンスのすべてではないが、大部分をマッピングすることができるのに対し、別の非効率なインデックス・フォーマットは少数の残りのシーケンスを処理できるときに利用すると、便利である。チェーン・メカニズムがないと、インデックス・シーケンスのすべてに非効率のフォーマットを使用しなければならないことになる。チェーン・フォーマットは変種と許容レベルが異なるたびに、異なるサブテーブルが必要になるときにも便利である。チェーン内の各サブテーブルは、現在取り扱われているマッピング許容範囲と変種に基づいてそれを含めるか、除外させるビット・フラグをもっている。本発明の第２、第３および第４側面によれば、ルックアップ・ハンドラ４１２がターゲット・コード化のマッピング・テーブル４１４をサーチするときは、含められたサブテーブルだけが考慮される。
【００９９】
さらに、本発明の第１から第４までの側面によれば、各サブテーブルに関連するこれらのビット・フラグはサブテーブル・マスクを形成している。また、呼び出し側要求（例えば、コード化の変種と許容範囲）と判断された属性（例えば、解明された方向とコンテキスト）は選択マスクを形成している。サブテーブル・マスクと選択マスク内のビット割当ては同じである。従って、特定のサブテーブルを含めるかどうかの判断はサブテーブル・マスクと選択マスクのビットワイズＡＮＤをとり、そのあと結果をサブテーブル・マスクと比較することによって行われる。ビットワイズＡＮＤの結果がサブテーブルのサブテーブル・マスクと同じであれば、そのサブテーブルが含められる。そうでなければ、それは含められない。
【０１００】
マッピング・テーブル４１４のヘッダ５００は好ましくは次のものを収めている。
・　一般的識別情報－フォーマット、長さ、チェックサムおよびバージョン。
・　最小ターゲット文字サイズ（バイト数）（文字サイズともいう）。
・　一般的フラグ（例えば、そのルップアップ・テーブルが方向またはコンテキスト・データを必要としているかどうか）。

・　そのテーブルによって処理される最大入力シーケンス長、および１からこの最大長までの入力シーケンス長を処理するテーブルを指定しているオフセット／長さのペアのリスト。
・　そのユニコードからのマッピングのデフォルト・フォールバック文字または文字シーケンス。
・　そのテーブルによってサポートされる変種のカウントとリスト。各変種ごとに、１つまたは２つ以上の関連ビット・マスクが指定される。単一の変種に複数のビット・マスクがある場合は、属性情報（方向、コンテキスト、および垂直形式の要求）はどのビット・マスクが使用されるかを判断するために使用される。ビット・マスクで「１」にセットされたビットは異なる変種をサポートするために種々のサブテーブルをオンにするために使用される。
・　可能とされる４つの許容範囲設定値（厳格／緩和、フォールバック・オン／オフ）の各々に関連する追加のビット・マスクのセット。該当する許容範囲レベル・マスクは変種マスクとＯＲがとられて、サブテーブルを使用可能または使用禁止にするために使用されるビット・マスクを形成する。

【０１０１】

Ｃ．コード変換処理

以下では、ユニコード・コード変換システム４００の好適実施例によって実行される処理について詳しく説明する。

【０１０２】

図６Ａはユニコード・コード変換システム４００を利用するアプリケーション（つまり、呼び出し側アプリケーション・プロセスまたはプログラム）によって実行される処理６００を示すフローチャートである。具体的には、図６Ａはユニコードからの処理に関係しているが、当然に理解されるように、類似のオペレーションを実行して他方の方向に変換することも可能である（ユニコードへの処理）ユニコードからのコンバータ４０２は全体的変換プロセスを制御する。

【０１０３】

最初に、処理６００は新しい状態インスタンスと変換のための制御情報を作成し、初期化する（６０２）。処理６００はインスタンスを設定するので、複数のスキャン・オペレーションを進行中にし、そのインスタンスによって区別することが可能である。次に、切り捨てが必要であるかどうかの判断６０４が行われる。切り捨てが必要な場合には、トランケータ４０７が起動される（６０６）。切り捨てが必要になるのは、入力データ・ストリームが、変換のためのデータを収めている受入れバッファ４０５の容量を越えたときである。切り捨てが必要でなければ、あるいは切り捨てが必要な場合には、ブロック６０６に続いて、ユニコードからのコンバータ４０２がコールされ（６０８）ユニコード文字列４０４を変換する。ユニコードからのコンバータ４０２の機能は、テキスト要素を取得し、そのためのターゲット・マッピングを調べることである。これについては、以下で詳しく説明する。変換機能から戻ると、処理６００はユニコードからのコンバータ４０２からターゲット文字列４０６を受け取る。

【０１０４】

次に、処理６００は変換が失敗したかどうかを判断する。変換が失敗していれば、エラーが処理される（６１４）。他方、変換が正常に行われていれば、変換が完了したかどうかの判断６１６が行われる。ユニコード文字列４０４の文字がターゲット・コード化に変換されたとき変換は完了する。変換が完了していれば、処理６００は完了し、ターゲット文字列４０６はコード変換を要求したプロセスまたはアプリケーションに利用可能にされる。さらに、処理６００は状態インスタンスと変換のための制御情報を破棄する（６１８）。他方、変換がまだ完了していないと判断６１６されたときは、処理６００は変換が完了するか、エラーが起こるまでブロック６０４～６１６を繰り返す。

【０１０５】

図６Ｂは本発明の第４側面による切り捨て処理６２０を示すフローチャートである。切

り捨て処理６２０は図６Ａのブロック６０６で起動され、トランケータ４０７によって実行される。

【０１０６】

切り捨て処理６２０は出力長をゼロ（０）に初期化する（６２２）。出力長はバッファの実効長、つまり、バッファの切り捨てられた長さに一致している。次に、次のテキスト要素が取得される（６２４）。次のテキスト要素の取得（６２４）に関連する処理はスキャナ４０８によって行われるが、これについては図９Ａ～図９Ｃを参照して以下で詳しく説明する。次に、テキスト要素がバッファの物理的長さ（バッファ長）を越える可能性があるかどうかの判断６２６が行われる。テキスト要素がバッファ長を越える可能性がなければ、出力長はそのテキスト要素を含むように更新される（６２８）。ここでは、実効長はテキスト要素がバッファ長を越える可能性がない限り、テキスト要素単位で大きくなる。ブロック６２８に続いて、考慮すべき追加テキストがバッファに残っているかどうかの判断６３０が行われる。考慮すべき追加テキストがバッファに残っていれば、処理はループしてブロック６２４に戻り、切り捨て処理６２０を繰り返す。他方、考慮すべき追加テキストがバッファに残っていないか、あるいはテキスト要素がバッファ長を越える可能性があるとブロック６２６で判断されていれば、出力長が戻される（６３２）。戻された出力長（６３２）はバッファの実効長（つまり切り捨てられた長さ）であるので、バッファは実効的にテキスト要素の終わり（つまり、バッファ内のテキストの最後のテキスト要素）で終了する。

【０１０７】

以上のようにして、切り捨て処理はバッファの実効長（つまり、切り捨てられた長さ）を判断する。これはバッファの切り捨てられた部分とも呼ばれる。切り捨ての後バッファに残っている余分のテキストは残余部分と呼ばれる。残余部分は入力ソース文字列の次のバッファ部分にキャリーオーバされ、その部分と一緒に考慮される。この処理はテキスト要素が正しく判断されることを保証する。

【０１０８】

以下は切り捨て処理の使い方の例である。

例：“．．．ＡＢＣＤ｀ＥＦＧ．．．”

バッファに置かれている部分が“Ｄ”の直後で終わっていれば、切り捨て処理はバッファに置かれている部分を切り捨てて、切り捨てられた長さが“Ｃ”のあとで終わるようにする。バッファに置かれた部分を切り捨てる必要があるのは、そのようにしないと、テキスト要素“Ｄ”がそのあとに置かれた結合マーク“｀”から切り離されることになるからである。切り離されると、テキストはターゲット・コード化に正しく変換されないことになる。残余部分は“Ｄ｀”であり、次の部分にキャリーオーバされ、“Ｄ｀ＥＦＧ．．”となる。

【０１０９】

図７はユニコード・コンバータ処理７００を示すフローチャートである。ユニコード・コンバータ処理７００は図６Ａのブロック６０８で実行されるオペレーションと関連づけられている。

【０１１０】

ユニコード・コンバータ処理７００は判断７０２から始まる。この判断７０２は変換すべきテキストがあるかどうかを判断する。変換すべきテキストがなければ、ユニコード・コンバータ処理７００は戻るだけである（または完了する）。他方、変換すべきテキストがあれば（つまり、ユニコード文字列４０４が完全に処理されていなければ）、処理７００が続けられる。まず、オフセットがオフセット配列に対して更新される（７０４）。オフセット配列入力文字列に関連するオフセット（ポインタ）の配列であり、フォント変更、ライン中断、言語変更などのある種の変更が、呼び出し側アプリケーションが有意と判断している入力文字列４０４内のどこで行われたかを示している。オフセット配列の更新７０４は、異なる文字長に合わせてオフセット（ポインタ）を調節することにより行われ

る。例えば、入力ユニコード文字列４０４のユニコード文字は２バイト長であるのに対し、ＡＳＣＩＩのターゲット・コード化に関連する文字のサイズは１バイト長である。ここでは、オフセット配列の更新７０４はオフセットがターゲット・コード化内の対応する文字を指すようにオフセットを更新する。実際には、入力文字列内のオフセットはコード化が異なっている出力文字列にマッピングされる。次に、次のテキスト要素が取得される（７０６）。スキャナ・テーブル４１０を用いるスキャナ４０８はユニコード文字列４０４からのテキスト要素を判断する。次のテキスト要素の取得７０６は以下で詳しく説明する。次に、取得されたテキスト要素はマッピング・テーブル４１４で調べられ（７０８）、ターゲット・コード化内のテキスト要素の変換コードが得られる。このルックアップはマッピング・テーブル４１４を使用してルックアップ・ハンドラ４１２によって行われる。変換コートのルックアップ７０８についても以下で詳しく説明する。

【０１１１】

次に、テキスト要素の変換コードが見つかったかどうかの判断７１０が行われる。変換コードが見つかったときは、ユニコード文字列４０４とターゲット文字列４０６の入力位置ポインタと出力位置ポインタがそれぞれ更新される（７１２）。入力位置ポインタは入力文字列４０４がどれだけ変換されたかを示している。出力位置ポインタはターゲット文字列４０６の長さを示している。ブロック７１２に続いて、処理７００はユニコード・コンバータ処理７００の先頭に戻り、変換すべきユニコード文字列４０４の次のテキスト要素（存在する場合）が処理できるようにする。

【０１１２】

しかるに、変換コードがマッピング・テーブル４１４に見つからないと判断７１０が判断したときは、呼び出し側（例えば、呼び出し側アプリケーション）がフォールバック処理を要求していたかどうかの判断７１４が行われる。呼び出し側がフォールバック処理を要求していれば、フォールバック処理が実行される（７１６）。フォールバック処理はフォールバック・ハンドラ４１６によって実行される。これは以下で詳しく説明する。他方、呼び出し側がフォールバック処理を要求していなければ、テキスト要素がルックアップ・ハンドラ４１２によってターゲット・コード化に変換できなかったのでエラーが通知される（７１８）。ブロック７１６と７１８に続いて、ユニコード文字列４０４とターゲット文字列４０６の入力位置ポインタと出力位置ポインタがそれぞれ更新され（７０２）、そのあと処理はユニコード・コンバータ処理７００の先頭に戻り、変換すべきユニコード文字列４０４０の次のテキスト要素（存在する場合）が処理できるようにする。

【０１１３】

図８はオフセット更新処理８００を示すフローチャートである。オフセット更新処理８００はオフセット配列が更新される図７のブロック７０４と関連している。

【０１１４】

オフセット更新処理８００は現在入力位置がオフセット配列にあるかどうかを判断８０２することから始まる。現在入力位置がオフセット配列にあれば、オフセット配列の内容がブロック８０４に続いて現在出力位置長さに従って更新される（８０４）、オフセット更新処理８００から戻る。他方、現在入力位置がオフセット配列にないときは、更新すべきオフセットがないのでオフセット更新処理８００から戻る判断８０２が行われるだけである。

【０１１５】

図９Ａおよび図９Ｂは本発明の第１、第２および第４側面による次のテキスト要素処理９００を示すフローチャートである。次のテキスト要素処理９００は次のテキスト要素を取得するとき図７のブロック７０６で実行されるオペレーションを具体化したものである。好ましくは、次のテキスト要素処理９００はスキャナ・テーブル４１０を使用してスキャナ４０８によって行われる。

【０１１６】

次のテキスト要素処理９００は状態と再配列フラグを初期化（９０２）することから始まる。次に、マッピング・テーブル４１４が方向情報を必要としているかどうかの判断９

０４が行われる。マッピング・テーブル４１４が方向情報を必要としていれば、ユニコード文字列４０４の方向が解明される（９０６）。本発明の第２側面では、マッピング・テーブル４１４が方向情報を必要としていれば、ユニコード文字列４０４の次の入力文字の方向が解明される（９０６）。方向が解明されると（９０６）、あるいは方向が必要でない場合には判断９０４の後で、文字のコンテキストに基づく判断９０８が行われる。ユニコード文字列４０４内の文字のコンテキストがコード変換（マッピング）に影響する可能性があるときは、コンテキストが解明される（９１０）。ブロック９１０に続いてまたは判断９０８に続いて、コンテキストが重要でないときは、ユニコード文字がユニコード文字列４０４から取得される（９１２）。ここでは、ユニコード文字列４０４内の次の文字が取得される。次に、取得されたユニコード文字の属性が調べられる（９１４）。属性ルックアップ９１４は図１１～図１３を参照して以下で詳しく説明する。次に、次のテキスト要素処理９００のアクションが判断される（９１６）。アクションの判断９１６は図１４Ａ、図１４Ｂおよび図１５を参照して以下で詳しく説明する。

【０１１７】

次に、アクションが“ＥＮＤ”であるかどうかの判断９１８が行われる。アクションが“ＥＮＤ”でなければ、アクションは少なくとも“ＡＤＶＡＮＣＥ”である。アクションが“ＡＤＶＡＮＣＥ”であるときは、アクションが“ＭＡＲＫ”でもあるかどうかの判断９２０が行われる。アクションが“ＭＡＲＫ”でもあるときは、文字がテキスト要素に挿入される（９２２）。さらに、“ＭＡＲＫ”アクションがとられると双方向状態がセーブされる（９２４）。双方向状態は方向性埋め込みスタックと双方向ステート・マシンの現在状態を含んでいる。ブロック９２４に続いてまたはアクションが“ＭＡＲＫ”でもないときは判断ブロック９２０に続いて、アクション修飾子に基づいてスイッチ・オペレーションが実行される。アクション修飾子はアクションの一部であり、“Ｓ”、“ＩＳＳ”、“ＡＳＳ”などの修飾子を含んでいる。アクションはどのアクション修飾子ももたないこともできる。アクション修飾子が“Ｓ”であるときは、再配列フラグがセットされる（９２８）。再配列フラグ（セットされたときは）はテキスト要素内の文字を再配列する必要がないことを示している。アクション修飾子が“ＩＳＳ”（つまり、シメトリック・スワッピング禁止）であるときは、スワップ・フラグがオフにセットされる（９３０）。アクション修飾子が“ＡＳＳ”（つまり、スワッピング禁止活動化）であるときは、スワップ・フラグがオンにセットされる（９３２）。スワップ・フラグはシメトリック・スワッピングが必要かどうかを示している。スイッチ９２６は拡張エリア９３４を使用して追加のアクション修飾子を含むように容易に適応させることができる。拡張エリア９３４を使用すると、ユーザはスキャナ４０８の振舞を変更することができる。また、アクション修飾子がなければ、次のテキスト要素処理９００はアクション修飾子に関連するどのオペレーションも実行しない。アクション修飾子オペレーションに続いて、現在文字インデックスが更新される（９３６）。現在文字インデックスは次のテキスト要素処理９００を実行するときソース文字列をスキャンしていくために使用されるソース文字列を指すポインタである。次に、次のテキスト要素処理９００はブロック９０４から始まるオペレーションを繰り返す。処理はアクションが“ＥＮＤ”であると判断９１８が判断するまでループしてブロック９０４－９３６を繰り返す。アクションが“ＥＮＤ”であると、判断９１８は判断ブロック９３８を実行させる。判断ブロック９３８は再配列フラグがセットされているかどうかを判断する。再配列フラグがセットされていれば（ブロック９２８）、テキスト要素内の文字は再配列される（９４０）。再配列は異種の文字クラスの加重値を提供する優先順位属性を使用して行うことが好ましい。再配列フラグがセットされていない場合にはブロック９３８に続いて、再配列フラグがセットされている場合にはブロック９４０に続いて、次のテキスト要素処理９００は完了し、ユニコード・コンバータ処理７００に戻る。

【０１１８】

図１０Ａ、図１０Ｂおよび図１０Ｃは本発明の第３側面による次のテキスト要素処理９００を示すフローチャートである。次のテキスト要素処理９００は次のテキスト要素を取

得するとき図７のブロック７０６によって実行されるオペレーションを具体化したものである。好ましくは、次のテキスト要素処理９００はスキャナ・テーブル４１０を使用してスキャナ４０８によって実行される。

【０１１９】

次のテキスト要素処理９００は状態と再配列フラグを初期化（９５２）することから始まる。次に、マッピング・テーブル４１４が方向情報を必要としているかどうかの判断９５４が行われる。マッピング・テーブル４１４が方向情報を必要としていれば、ユニコード文字列４０４の次の入力文字の方向が解明される（９５６）。次に、ユニコード文字がユニコード文字列４１４から取得される（９５８）。ここでは、ユニコード文字列４０４内の次の文字が取得される（９５８）。次に、取得されたユニコード文字の属性が調べられる（９６０）。属性ルックアップ９６０は図１１～図１３を参照して以下で詳しく説明する。次に、次のテキスト要素処理９００のアクションと次の状態が判断される（９６２）。アクションと次の状態の判断は図１４Ａ、図１４Ｂ、図１６Ａおよび図１６Ｂを参照して以下で詳しく説明する。

【０１２０】

次に、アクションが“ＥＮＤ”であるかどうかの判断９６４が行われる。アクションが“ＥＮＤ”でなければ、アクションは少なくとも“ＡＤＶＡＮＣＥ”である。アクションが“ＡＤＶＡＮＣＥ”であるときは、アクションが“ＭＡＲＫ”でもあるかどうかの判断９６６が行われる。アクションが“ＭＡＲＫ”でもあるときは、文字がテキスト要素に挿入される（９６８）。さらに、“ＭＡＲＫ”アクションがとられると双方向状態がセーブされる（９７０）。双方向状態は方向性埋め込みスタックと双方向ステート・マシンの現在状態を含んでいる。ブロック９７０に続いてまたはアクションが“ＭＡＲＫ”でもないときは判断ブロック９６６に続いて、アクション修飾子に基づいてスイッチ・オペレーション９７２が実行される。アクション修飾子はアクションの一部であり、“Ｓ”、“ＩＳＳ”、“ＡＳＳ”などの修飾子を含んでいる。アクションはどのアクション修飾子ももたないこともできる。アクション修飾子が“Ｓ”であるときは、再配列フラグがセットされる（９７４）。再配列フラグ（セットされたときは）はテキスト要素内の文字を再配列する必要がないことを示している。アクション修飾子が”ＩＳＳ”（つまり、シメトリック・スワッピング禁止）であるときは、スワップ・フラグがオフにセットされる（９７６）。アクション修飾子が”ＡＳＳ”（つまり、スワッピング禁止活動化）であるときは、スワップ・フラグがオンにセットされる（９８０）。スワップ・フラグはシメトリック・スワッピングが必要かどうかを示している。スイッチ９７２は拡張エリア９８０を使用して追加のアクション修飾子を含むように容易に適応させることができる。拡張エリア９８０を使用すると、ユーザはスキャナ４０８の振舞を変更することができる。また、アクション修飾子がなければ、次のテキスト要素処理９００はアクション修飾子に関連するどのオペレーションも実行しない。アクション修飾子オペレーションに続いて、現在文字インデックスが更新される（９８２）。現在文字インデックスは次のテキスト要素処理９００を実行するときソース文字列をスキャンしていくために使用されるソース文字列を指すポインタである。次に、次のテキスト要素処理９００はブロック９５４から始まるオペレーションを繰り替えす。処理はアクションが“ＥＮＤ”であると判断９６４が判断するまでループしてブロック９５４～９８２を繰り返す。アクションが“ＥＮＤ”であると、判断９６４はコンテキスト処理９８４を実行させる。コンテキスト処理９８４のあと、判断ブロック９８６は再配列フラグがセットされているかどうかを判断する。再配列フラグがセットされていれば（ブロック９７４を参照）、テキスト要素内の文字は再配列される（９８８）。再配列は異種の文字クラスの加重値を提供する優先順位属性を使用して行うことが好ましい。再配列フラグがセットされていない場合にはブロック９８６に続いて、再配列フラグがセットされている場合にはブロック９８８に続いて、次のテキスト要素処理９００は完了し、ユニコード・コンバータ処理７００に戻る。

【０１２１】

次に、図１０Ｃを参照してコンテキスト処理９８４について説明する。コンテキスト処

理９８４はここで説明している実施例では、スキャナ・テーブル４１０を使用するスキャナ４０８によって実現されている。コンテキスト処理９８４は判断９８５から始まり、アクションがＥｎｄＯｕｔｐｕｔＸｎ（つまり、単独コンテキスト）であるかどうかが判断される。そうであれば、コンテキスト・マスクがセットされ（９８６）、コンテキストが単独であることを示し、コンテキスト処理９８４は完了し、戻る。他方、アクションがＥｎｄＯｕｔｐｕｔＸｎでなければ、アクションがＥｎｄＯｕｔｐｕｔＸ１（つまり、初期コンテキスト）であるかどうかが判断９８７で判断される。そうであれば、コンテキスト・マスクがセットされ（９８８）、コンテキストが初期であることを示し、コンテキスト処理９８４は完了し、戻る。他方、アクションがＥｎｄＯｕｔｐｕｔＸ１でなければ、アクションがＥｎｄＯｕｔｐｕｔＸｒ（つまり、終了コンテキスト）であるかどうかが判断９８９される。そうであれば、コンテキスト・マスクがセットされ（９９０）、コンテキストが初期であることを示し、コンテキスト処理９８４は完了し、戻る。他方、アクションがＥｎｄＯｕｔｐｕｔＸｒでなければ、アクションがｅｎｄＯｕｔｐｕｔＸｍ（つまり、中間コンテキスト）であるかどうかの判断９９１が行われる。そうであれば、コンテキスト・マスクがセットされ（９９２）、コンテキストが初期であることを示し、コンテキスト処理９８４は完了し、戻る。アクションがＥｎｄＯｕｔｐｕｔＸｍでないと判断９９１されたときは、テキスト要素は関連するコンテキストをもっていないので、コンテキスト・マスクは「無視」にセットされる（９９３）。

【０１２２】

以上のように、コンテキスト処理９８４は上記実施例では、テキスト要素とコンテキストが一緒に判断されるように実現されている。どちらの判断も、スキャナ４０８とスキャナ・テーブル４１０によって先読みスキャンニングを利用してテキスト要素が完成し、入力テキスト・ストリーム内のテキスト要素の内容が分かるようにしている。コンテキスト・マスクに入っているコンテキスト情報は、マッピング・テーブル４１４とやりとりして、判断されたコンテキストをもつテキスト要素の正しいターゲット・コード化を探すときにルックアップ・ハンドラ４１２によって使用される。コンテキスト処理のオペレーションの詳細は図１６Ａと図１６Ｂを参照して以下で説明する。

【０１２３】

図１１はスキャナ４０８を示すブロック図である。特に、スキャナ４０８は属性ハンドラ１０００とテキスト要素ハンドラ１００２を含んでいる。テキスト要素ハンドラ１００２は図９Ａと図９Ｂおよび図１０Ａ～図１０Ｃを参照して上述した次のテキスト要素処理９００を実行する。属性ハンドラ１０００は属性テーブル１００４とやりとりして、次のテキスト要素処理９００が必要とするユニコード文字の属性を取得する（図９Ａのブロック９１４と図１０Ａのブロック９６０を参照）。属性には次のものがある。つまり、方向、クラス、優先順位、シメトリック・スワッピング、サブセットおよびコンテキストである。方向属性は方向を解明するとき使用される（図９Ａのブロック９１４および図１０Ａと図１７Ａのブロック９５６を参照。クラス属性はアクション（例えば、ＡＤＶＡＮＣＥ、ＥＮＤ）を判断するためにスキャナ４０８によって使用される。優先順位属性はテキスト要素内の文字を記録するために使用される（図９Ｂのブロック９４０と図１０Ａのブロック９８８を参照）。シメトリック・スワッピング属性はシメトリック・スワッピングが必要であるかどうかを判断するために使用される。コンテキスト属性はコンテキストを解明するときに使用される（図９Ａのブロック９１０を参照）。

【０１２４】

図１２は図１１の属性テーブル１００４の好ましいフォーマットを示す概略図である。属性テーブル１００４はヘッダ部分１１００、範囲テーブル部分１１０２、および属性テーブル部分１１０４を含んでいる。ヘッダ部分１１００は次のものに関する情報を含んでいる。つまり、各テーブルの総テーブル・サイズ、チェックサム値、バージョン、オフセットおよび要素の数である。範囲テーブル部分１１０２内の要素は属性が共通にグループ化されている範囲を示し、各グループごとに、属性テーブル部分１１０４の該当部分を指すポインタをもっている。属性テーブル１００４の範囲テーブル部分１１０２のフォーマ

ットは、例えば、各エントリごとに、範囲値の開始、範囲値の終了、および範囲に関連するデータ・ワードを含んでいる。属性テーブル１００４の構成は属性情報のコンパクトなストアを容易にしている。別のストア構成も可能であるが、データ・ストアのコンパクトさから見たとき非効率的である。

【０１２５】

図１３は属性ルップアップ処理１２００を示すフローチャートである。属性ルックアップ処理１２００は属性ハンドラ１０００によって実行され、図９Ａのブロック９１４または図１０Ａのブロック９６０で次のテキスト要素処理９００によって開始される。

【０１２６】

属性ルックアップ処理１２００は属性テーブル１００４の範囲テーブル部分１１０２内の範囲を使用してバイナリ・サーチから開始される。該当の範囲がバイナリ・サーチで見つかると、その範囲に関連するデータ・ワードが範囲テーブル部分１１０２から取得される（１２０４）。好ましくは、データ・ワードの最初のビットは間接ビットである。次に、データ・ワードの間接ビットがセットされているかどうかの判断１２０６が行われる。データ・ワードの間接ビットがセットされていない場合は、データ・ワード自体は現在の文字の属性を収めているので、データ・ワードは属性として戻される（１２０８）。他方、データ・ワードの間接ビットがセットされていれば、属性は、範囲テーブル部分１１０２から取得されたデータ・ワードを属性テーブル部分１１０４までのインデックスまたはオフセットとして使用して属性テーブル部分１１０４から取得される。従って、この場合のデータ・ワードは範囲内の各文字の属性を収めている配列までのインデックスまたはオフセットである。ブロック１２０８または１２１０に続いて、属性ルックアップ処理１２００は完了し、戻る。

【０１２７】

図１４Ａと図１４Ｂは次のアクションを判断するためにスキャナ４０８によって使用されるスキャナ・テーブル１３００（４１０）に関連する概略図である。次のアクションの判断は次のテキスト要素処理９００（図９Ａ）のブロック９１６によって起動されるか、ブロック９６２（図１０Ａ）によって起動される。図１４Ａは本発明で使用されるスキャナ・テーブル１３００の好ましいフォーマットを示す図である。スキャナ・テーブル１３００は「現在状態」を１つのインデックスとして、「クラス」を別のインデックスとしてもつ２次元配列である。「クラス」はクラス属性を指している。これらのインデックスはスキャナ・テーブル１３００内の代表的な要素１３０２を選択する。図１４Ｂはスキャナ・テーブル１３００の代用的な要素１３０２を示す図である。代表的な要素１３０２はスキャナ４０８の次の状態を収めている次の状態部分１３０４を含み、アクション部分１３０６はスキャナ４０８のアクションを示している。実際には、スキャナ４０８はスキャナ・テーブル１３００と一緒になって、スキャナをどのように動作させるかを判断するステート・マシンを実現している。

【０１２８】

スキャナ・テーブル１３００は異種の文字コード化に関する異なる次の状態とアクションを収めている。

【０１２９】

本発明の第１、第２および第４側面によれば、図１５は好ましいレイアウトとスキャナ・テーブル１３００にストアされる情報の両方を示しているテーブル１４００である。このテーブルに関連する表記は次の通りである。

【０１３０】

文字クラス：

ＣＣ　－制御文字
ＯＳ　－他のスペース
ＮＳ　－非スペース
ＬＤ　－ラテン・ディジット
ＦＳ　－フラクション・スラッシュ

ＪＬ（ｆ）－ジャモス先頭子音（フィラー）
ＪＶ（ｆ）－ジャモス母音（フィラー）
ＪＴ　　－ジャモス・トレーラ
ＮＵ　　－有効なユニコード文字でない
ＩＳＳ　－シメトリック・スワッピング禁止
ＡＳＳ　－シメトリック・スワッピング活動化

【０１３１】

次の状態：
状態０　－終了。テキスト要素を戻すべきかどうかを、２重および半分音符号の状態に基づいて判断する。

【０１３２】

状態１　－開始状態
状態２　－非スペース（分音符号）の追加
状態３　－数値フラクションの有無チェック
状態７　－朝鮮語ジャモス

【０１３３】

アクション：
Ａｄｖ　　－［ＡＤＶＡＮＣＥ］次の文字へ進む（現在文字は現在テキスト要素（ＴＥ）に含まれている場合と含まれていない場合がある）
ＡｄｖＭａｒｋ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ］現在文字に最終文字のマークを付け次の文字へ進む
ＡｄｖＭａｒｋＳ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ＋Ｓ］現在文字に最終文字のマークを付け次の文字へ進み、再配列フラグをセットする
ＡｄｖＭａｒｋＡＳＳ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ＋ＡＳＳ］現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを活動化する
ＡｄｖＭａｒｋＩＳＳ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ＋ＩＳＳ］現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを禁止する
Ｅｎｄ　　－テキスト要素を最終のマークを付けた文字で終了する

【０１３４】

注意：すべての機能は再配列フラグがセットされているかをチェックして確かめ、非スペース文字を開始ポインタから始めて再配列する。再配列する必要のある非スペース・マークのセットは複数存在することがあるので、文字列全体をチェックするのが最良の方法である。もちろん、文字を再配列すると、再配列フラグはクリアされる。

【０１３５】

本発明の第３側面によれば、図１６Ａおよび図１６Ｂは好ましいレイアウトと情報の両方を示すテーブル１４００であり、情報はソース文字列内のテキスト要素とそのコンテキストの両方の判断を容易にするためにスキャナ・テーブル４１０にストアされるものである。このテーブルに関連する表記は次のとおりである。

【０１３６】

文字クラス：
ＣＣ　－制御文字
ＯＳ　－他のスペース
ＮＳ　－非スペース
ＬＤ　－ラテン・ディジット
ＦＳ　－フラクション・スラッシュ
ＤＤ　－二重分音符号

ＨＤ　－半分音符号
ＣＨ　－ジャモス先頭子音（フィラー）
ＪＯ　－ジャモス母音（フィラー）
ＪＶ　－ジャモス子音トレーラ
ＮＵ　－有効なユニコード文字でない
ＩＳＳ　－シメトリック・マッピング禁止
ＡＳＳ　－シメトリック・マッピング活動化
ＨＨ　－高ハーフゾーン
ＬＨ　－低ハーフゾーン
Ｖ　－Ｖｉｒａｍａ
ＺＷＮＪ　－ゼロ幅の非ジョインダ
Ｒ　－右リンク
Ｄ　－２重リンク
Ｃ　－リンクを引き起こす
Ｔ　－透過

【０１３７】

次の状態：
状態０　－終了。テキスト要素を戻すべきかどうかを２重分音符号と半分音符号に基づいて判断する
状態１　－開始状態
状態２　－非スペース（分音符号）追加
状態３　－ラテン・ディジット
状態４　－ラテン・ディジット・シーケンス
状態５　－ラテン・ディジット・シーケンスとそのあとに続くフラクション・スラッシュ
状態６　－ラテン・ディジット・シーケンス、フラクション・スラッシュ、ラテン・ディジット・シーケンス
状態７　－Ｃｈｏｓｅｏｎｇシーケンス
状態８　－Ｃｈｏｓｅｏｎｇシーケンスとそのあとに続く
Ｊｕｎｇｓｅｏｎｇシーケンス
状態９　－Ｃｈｏｓｅｏｎｇシーケンス、Ｊｕｎｇｓｅｏｎｇシーケンス、Ｊｏｎｇｓｅｏｎｇ
状態１０　－高半文字
状態１１　－高半文字とそのあとに続く低半文字
状態１２　－Ｖｉｒａｍａとそのあとに続くゼロ幅ジョインダまたはゼロ幅非ジョインダ
状態１３　－コンテキストの特殊開始
状態１４　－右リンク文字（特殊コンテキスト状態）
状態１５　－２重リンク文字（特殊コンテキスト状態）
状態１６　－リンクを引き起こす文字（通常または特殊状態）
状態１７　－右リンク文字（通常状態）
状態１８　－２重リンク文字（通常状態）

【０１３８】

次の開始状態：［実際は次の状態のサブセット］
開始１　－開始状態
開始１３　－コンテキストの特殊開始状態

【０１３９】

アクション：
Ａｄｖ　－［ＡＤＶＡＮＣＥ］次の文字へ進む（現在文字は現在テキスト要素（ＴＥ）に含まれている場合と含まれていな

い場合がある）
ＡｄｖＭａｒｋ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ］現在文字に最終文字のマークを付け次の文字へ進む
ＡｄｖＭａｒｋＳ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ＋Ｓ］現在文字に最終文字のマークを付け次の文字へ進み、再配列フラグをセットする
ＡｄｖＭａｒｋＡＳＳ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ＋ＡＳＳ］現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを活動化する
ＡｄｖＭａｒｋＩＳＳ　－［ＡＤＶＡＮＣＥ＋ＭＡＲＫ＋ＩＳＳ］現在文字に最終文字のマークを付け次の文字へ進み、シメトリック・マッピングを禁止する
Ｅｎｄ　－テキスト要素を最終のマークを付けた文字で終了する
ＥｎｄＯｕｔｐｕｔＸｎ　－［ＥＮＤ＋単独コンテキスト］テキスト要素を終了し、単独コンテキストであることを示す
ＥｎｄＯｕｔｐｕｔＸ１　－［ＥＮＤ＋初期コンテキスト］テキスト要素を終了し、初期コンテキストであることを示す
ＥｎｄＯｕｔｐｕｔＸｒ　－［ＥＮＤ＋終了コンテキスト］テキスト要素を終了し、終了コンテキストであることを示す
ＥｎｄＯｕｔｐｕｔＸｍ　－［ＥＮＤ＋中間コンテキスト］テキスト要素を終了し、中間コンテキストであることを示す

【０１４０】
注意：すべての機能は再配列フラグがセットされているかをチェックして確かめ、非スペース文字を開始ポインタから始めて再配列する。再配列する必要のある非スペース・マークのセットは複数存在することがあるので、文字列全体をチェックするのが最良の方法である。もちろん、文字を再配列すると、再配列フラグはクリアされる。

【０１４１】
以下では、スキャナ・テーブル１４００の使用例を３つ示して説明する。最初の２つの例は本発明の第１、第２、第３および第４側面に関係し、最後の例は第３側面に関係するものである。

【０１４２】
例１：入力文字列“ＡＡＢ”
文字クラスは３文字ともＯＳである。最初の文字“Ａ”が取得される。開始状態（状態１）から始まり、最初のアクションはＡｄｖＭａｒｋ、次の状態は状態２である。これにより、最初の文字“Ａ”が現在テキスト要素内に挿入され、次の文字（２番目の文字“Ａ”）が取得される。次に、状態２では、アクションはＥｎｄ、次の状態は状態０である。従って、テキスト要素は最初のテキスト要素だけを含んでいる。同じシーケンスはこの特定入力文字列の２番目と３番目の文字について繰り返される。このようにして、入力文字列の文字の各々は分離しているが、隣接しているテキスト要素に割り当てられる。

【０１４３】
例２：入力文字列“Ａ｀Ｂ”
文字クラスは入力文字列の最初と最後の文字ではＯＳである。２番目の文字の文字クラスがＮＳであるのは、これが結合マークであるためである。最初の文字“Ａ”が取得される。開始状態（状態１）から始まり、最初のアクションはＡｄｖＭａｒｋ、次の状態は状態２である。これにより、最初の文字“Ａ”が現在テキスト要素内に挿入され、次の文字（２番目の文字“｀”）が取得される。次に、状態２では、アクションはＡｄｖＭａｒｋＳ、次の状態は状態２である。これにより、２番目の文字“｀”が現在テキスト要素に挿入される。次に、３番目の文字が取得される。この時点の状態２のアクションはＥｎｄ、次の状態は状態０である。従って、テキスト要素は入力文字列の１番目と２番目の文字を含んでいる。３番目の文字は例１の場合と同じように、自身のテキスト要素に置かれる。

【０１４４】
例３：入力文字列“ＯＳ　Ｒ　Ｄ　ＯＳ　ＯＳ”［文字間にスペースがなく、各文字はその文字クラスで表されている。ＲとＤ文字クラスはコンテキスト・ベースの表示形態をもつ文字を含んでいるが、ＯＳ文字クラスは含んでいない。］
【０１４５】
最初の文字は文字クラスＯＳである。開始状態（状態１）から始まり、最初のアクションはＡｄｖＭａｒｋ、次の状態は２、次の開始状態は１である。これにより、最初の文字が現在テキスト要素に挿入され、２番目の文字が取得される。２番目の文字は文字クラスＲである。次に、状態２では、アクションはＥｎｄ、次の状態は０、次の開始状態は１である。従って、最初のテキスト要素は最初の文字だけを含み、コンテキスト・フラグは無視にセットされる。
【０１４６】
次のテキスト要素を判断するときは、処理は２番目の文字から始まり、新しい開始状態は１になっている。この時点で、スキャナ・テーブルから得られるアクションはＡｄｖＭａｒｋであり、次の状態は１７、次の開始状態は１３である。これにより、２番目の文字が現在テキスト要素に挿入され、３番目の文字が取得される。３番目の文字は文字クラスＲである。次に、状態１７では、アクションはＥｎｄＯｕｔｐｕｔＸｎ、次の状態は０、次の開始状態は１である。従って、２番目のテキスト要素は２番目の文字だけを含み、コンテキストはＸｎ（単独）である。
【０１４７】
次のテキスト要素を判断するときは、処理は３番目の文字から始まり、新しい開始状態は１３になっている。この新しい開始状態は最後の文字についてテーブルで示された次の開始状態である。ここでは、スキャナ・テーブルから得られるアクションはＡｄｖＭａｒｋであり、次の状態は１５、次の開始状態は１３である。３番目の文字は現在テキスト要素の一部であるので、４番目の文字が取得される。４番目の文字は文字クラスＯＳである。次に、状態１５では、アクションはＥｎｄＯｕｔｐｕｔＸｒ、次の状態は０、次の開始状態は１である。従って、３番目のテキスト要素は３番目の文字だけを含み、コンテキストはＸｒ（終了）である。
【０１４８】
次のテキスト要素を判断するときは、処理は４番目の文字から始まり、新しい開始状態は１３になっている。スキャナ・テーブルから得られるアクションはＡｄｖＭａｒｋであり、次の状態は２、次の開始状態は１である。これにより、４番目の文字が現在テキスト要素に挿入され、５番目の文字が取得される。５番目の文字は文字クラスＯＳである。次に、状態２では、アクションはＥｎｄ、次の状態は０、次の開始状態は１である。従って、４番目のテキスト要素は４番目の文字だけを含み、コンテキスト・フラグは無視にセットされる。
【０１４９】
図１７Ａは本発明の第２側面による方向解明処理１５００を示すフローチャートである。方向解明処理１５００は次のテキスト要素処理９００（図９Ａ）内のブロック９０６で実行される処理である。方向解明処理１５００は、スキャナ４０８のために解明された方向の状態が初期状態にあるかどうかを判断する判断１５０２から開始される。初期状態では、スキャナ４０８はテキスト要素内の文字の方向を知らない。スキャナ４０８のために解明された方向の状態が初期状態にあれば、入力文字列の初期方向が判断される（１５０４）。初期状態１５０４の判断１５０４に関連する処理は図１８を参照して以下で詳しく説明する。他方、スキャナ４０８のために解明された方向の状態が初期状態にないときは、初期方向を判断する必要はない。どちらの場合も、ブロック１５０２またはブロック１５０４に続いて、ユニコード文字が入力ストリームから取得される（１５０６）。次に、ユニコード文字に関連する属性が調べられる（１５０８）。方向属性は方向解明処理１５００を行うとき重要な属性である。方向解明処理１５００のブロック１５０６と１５０８が実行するオペレーションは次のテキスト要素処理９００のブロック９１２と９１４のそ

れと同じであるので、これ以上詳しく説明することは省略する。

【０１５０】

次に、方向、次の状態およびアクションが判断される（１５１０）。方向、次の状態およびアクションは方向属性と現在状態を使用して判断され、テーブル属性ルップアップ・プロセスを使用して方向テーブルから取得される。方向テーブルは方向属性と現在状態によってインデックスされる２次元配列である。インデックスが指している方向テーブル内の要素は文字の方向、次の状態、および方向解明処理１５００がとるべきアクションを含んでいる。方向は次のものの１つである。すなわち、左、右、グローバル、およびＮＯ＿ＯＵＴＰＵＴである。取り得るアクションは、ＮＯ＿ＡＣＴＩＯＮ、ＰＵＳＨ　ＲＯ、ＰＵＳＨ　ＲＥ、ＰＵＳＨ　ＬＥ、ＰＵＳＨ　ＬＯ、ＰＯＰ、およびＲＥＳＥＴである。方向は明示的オーバライド文字により入力文字列の中で変化することがあるので、以前の方向は方向スタック上にストアされている。従って、「プッシュ」および「ポップ」の使用はこの分野で周知であるスタック操作コマンドを意味する。“ＲＯ”は右から左へのオーバライドを意味し、“ＬＯ”は左から右へのオーバライドを意味し、“ＲＥ”は右から左への埋め込みを意味し、“ＬＥ”は左から右への埋め込みを意味する。好ましくは、方向テーブルと方向解明処理１５００はステート・マシンとして動作する。ステート・マシンは基本的にユニコード標準バージョン１．０　６１１－６２１ページ（付録Ａ）に記載されている双方向アルゴリズムに従って動作するが、１回のパスだけで結果が得られるのに対し、ユニコード標準に記載されているアルゴリズムは複数のパスを必要とする。

【０１５１】

図１７Ｂ～図１７Ｄは本発明の第２側面による双方向アルゴリズムの好ましいインプリメンテーションによる双方向状態テーブル１５１１を示している。双方向状態テーブル１５１１はテーブル駆動型ステート・マシンを実現している。各カラムは単一状態である。状態はそこに記録される情報を示唆する名前が付けられている。各行は図１７Ｂ～図１７Ｄにまたがっており、以下に示す文字クラス名の１つが付けられている。

【０１５２】

ＬＲ　左から右へが主流
ＲＬ　右から左へが主流
ＡＬ　アラビア文字（右から左へが主流）
ＬＲＥ　左から右への埋め込みマーク
ＲＬＥ　右から左への埋め込みマーク
ＬＲＯ　左から右へのオーバライド・マーク
ＲＬＯ　右から左へのオーバライド・マーク
ＰＤＦ　ポップ方向フォーマット・マーク
ＡＮ　アラビア数字
ＥＮ　ヨーロッパ数字
ＥＴ　ヨーロッパ数字ターミネータ
ＥＳ　ヨーロッパ数字セパレータ
ＣＳ　共通数字セパレータ
ＯＮ　その他の中立文字
ＢＳ　ブロック・セパレータ

【０１５３】

双方向状態テーブル１５１１の各セルは関連のアクションと出力を持つ遷移を表している。これらのセルは新しい状態の名前、とられるオプションのアクション、および出力（もしあれば）を収めている。新しい状態は現在のグローバル方向に左右される場合がある。これは新しい状態名に（Ｇ）を入れることで示される。グローバル方向が未知のときこれらの遷移の１つを行うとエラーになる。取り得るアクションは次のとおりである。

【０１５４】

プッシュ　新しい埋め込み状態を埋め込みスタック上にプッシュする。プッシュされる実際の値は現れた実際の埋め込み制御によって決ま

る。このインプリメンテーションでは、これを処理するアクション動詞は４つある。

【０１５５】

ポップ　　現在の埋め込み状態をスタックからポップし、スタックの新しいトップを現在の埋め込み状態にする。新しい埋め込みがオーバライドであれば、セルに入っているターゲット状態にではなくＯＲ状態に遷移が行われる。

【０１５６】

リセット　埋め込み状態をクリアし、文字を消費することなくＳＴＡＲＴに移る。すなわち、エプシロン遷移を行う。リセットには出力はない。

【０１５７】

エラー　　即時エラーを引き起こす。

【０１５８】

出力はそれぞれ左から右へと右から左へを意味するＬまたはＲのどちらか、現在のグローバル方法の出力を意味するＧ、または出力なしを意味する＊である。マシンには、小文字の‘ｓ’で始まらない名前を持つ状態で入ることができる。ＳＴＡＲＴ状態は新しいテキスト・ブロックを目的としている。ｓＤＩＲ状態はグローバル方向を判断するときのエントリ・ポイントとして使用される。この計算はメイン・スキャンと同時に行うことが可能であるが、別々にすると単純化される。

【０１５９】

図１５Ａに戻って説明すると、ブロック１５１０に続いて、アクションが“ＲＥＳＥＴ”であるかどうかの判断１５１２が行われる。アクションが“ＲＥＳＥＴ”であれば、処理は方向解明処理１５００の先頭に戻る。そうでなければ、方向解明処理１５００が続行される。つまり、判断１５１２に続いて、アクションが実行される（１５１４）。

【０１６０】

次に、方向（ブロック１５１０で判断されたもの）が“ＮＯ＿ＯＵＴＰＵＴ”であるかどうかの判断１５１６が行われる。方向が“ＮＯ＿ＯＵＴＰＵＴ”に等しくなければ、方向がコンテキスト（各インスタンスごとに）にセットされ（１５１８）、状態が保存される（１５２０）。ブロック１５２０に続いて、方向解明処理１５００は完了し、戻る。

【０１６１】

しかるに、出力が“ＮＯ＿ＯＵＴＰＵＴ”であると判断１５１６で判断されたときは、入力文字列全体が処理されたかどうかの判断１５２２が行われる。入力文字列全体が処理されていなければ、処理は方向解明処理１５００の先頭に戻り追加のユニコード文字を取得し、処理できるようにするが、これはテキスト要素の方向がまだ判断されていないためである。他方、入力文字列全体が処理されたと判断１５２２されたときは、処理がテキストの終わりまで達したかどうかが判断１５２４される。つまり、ターゲット・コード化に変換すべきテキストが方向解明処理１５００によって完全に処理されたかどうかが判断される。変換すべき追加テキストがあれば、入力文字列の現在の文字では方向が計算できなかったのでエラーになる（１５２８）。しかし、追加テキストがなければ、ブロック・セパレータの方向がブロックの全体方向から判断される（１５２６）（１５０４を参照）。ブロック１５２６に続いて、ブロック１５１８と１５２０が前述したように実行され、そのあと方向解明処理１５００は完了し、戻る。

【０１６２】

図１８は本発明の第２側面による初期方向判断処理１６００を示すフローチャートである。初期方向判断処理１６００は図１７Ａのブロック１５０４で実行されるオペレーションと関連している。

【０１６３】

初期方向判断処理１６００は制御フラグに基づくスイッチ・オペレーションから開始される。制御フラグは次の１つを示している。すなわち、

ＮＯ＿ＯＵＴＰＵＴ、Ｌ－ｔｏ－ＲまたはＲ－ｔｏ－Ｌである。これらの制御フラグはユニコード・コード変換システム４００を起動するアプリケーションによってセットされる（つまり、制御フラグはコンバータへの入力である）。制御フラグがＲ－ｔｏ－Ｌを示しているときは、グローバル方向はＲ－ｔｏ－Ｌにセットされる（１６０４）。制御フラグがＬ－ｔｏ－Ｒにセットされているときは、グローバル方向はＬ－ｔｏ－Ｒにセットされる（１６０６）。制御フラグがＮＯ＿ＯＵＴＰＵＴにセットされているときは、スモール・ループが開始され、方向が判断できるまで入力文字列のユニコード文字をスキャンしていく。ループは状態を“ＳＴＡＲＴ　ＳＴＡＴＥ”にセット（１６０８）することから開始される。次に、ユニコード文字が入力文字列から取得される（１６１０）。ユニコード文字の属性が次に調べられる（１６１２）。属性は図９Ａのブロック９１４で使用され、図１２に詳しく説明されているものと同じ方法を用いて調べられる（１６１２）。次にユニコード文字の方向が属性（つまり、方向属性）を使用して判断される（１６１４）。方向が“ＮＯ＿ＯＵＴＰＵＴ”に等しいかどうかがの判断１６１６が行われる。方向が“ＮＯ＿ＯＵＴＰＵＴ”であれば、入力文字列の終わりまで達したかどうかの判断１６１８が行われる。入力文字列の終わりまで達していなければ、処理は戻り、ブロック１６１０～１６１８を繰り返す。文字列の終わりまで達していると、判断１６１８は方向を見つけることなく特殊方向スキャニング・ループを終わらせる（例えば、ＮＯ＿ＯＵＴＰＵＴ）。そうでなければ、スキャニング・ループは判断１６１６が方向を判断したとき終了する。

【０１６４】

いずれの場合も、スイッチ・オペレーション１６０２に関連する方向処理に続いて、スイッチ・オペレーション１６２０は判断に基づいて起動される。判断された方向が２ＮＯ＿ＯＵＴＰＵＴ”であるときは、現在レベルは埋め込みボトムにセットされる（１６２２）。他方、方向がＬ－ｔｏ－Ｒであれば、現在レベルはゼロにセットされ、前のレベルはボトムにセットされ、オーバライド状況は中立にセットされる（１６２４）。方向がＲ－ｔｏ－Ｌであるときは、現在レベルは１にセットされ、前のレベルはボトムにセットされ、オーバライド状況は中立にセットされる（１６２６）。レベルとは、ユニコード標準に記載されている埋め込みレベルのことである。初期方向判断処理１６００の特殊方向スキャニング・ループ（例えば、１６１０～１６１８）は図１８に別ループとして示されているが、総システム処理効率は特殊方向スキャニング・ループをメイン方向スキャニング・ループ内に組み入れると向上することができる。

【０１６５】

図１９はテキスト要素ルックアップ処理１６３０を示すフローチャートである。テキスト要素ルックアップ処理１６３０はルックアップ・ハンドラ４１２によって実行され、ユニコード・コンバータ処理７００（図７）内のブロック７０８によって起動される。

【０１６６】

テキスト要素ルックアップ処理１６３０は変種リストをサーチし、実際の属性と要求された変種に一致するエントリを見つけることから開始される。変種のサーチ１６３２は図２４を参照して以下で詳しく説明する。次に、一致するものが見つかったかどうかの判断１６３４が行われる。一致するものが見つからなければ、エラーになる（１６３６）。他方、一致するものが見つかれば、対応するビット・マスクが変種リストから取得される（１６３８）。好ましくは、変種リストは３つのフィールド、つまり、変種識別子、属性のセット、およびビット・マスクをもっている。変種リスト内の変種識別子と属性セットが実際の属性および要求された変種と一致していれば、対応するビット・マスクが変種リストから選択される。ブロック１６３８に続いて、ビット・マスクは許容範囲ビット・マスクと結合され（１６４０）選択フラグが得られる。選択フラグは上述したようにマッピング・テーブル４１４のサブテーブルを選択するとき使用される選択マスクを形成する。好ましくは、本発明の第２側面では、結合１６４０はビットワイズ・オペレーションである。好ましくは、本発明の第１、第３および第４側面では、結合１６４０はビットワイズＯＲオペレーションである。次に、現在テキスト要素の長さのためのテーブルがマッピング・テーブル４１４内にあるかどうかの判断１６４２が行われる。なければ、エラーになる

（１６４４）。他方、現在テキスト要素の長さのためにテーブルがあれば、ルックアップ・テーブルとそのフォーマットが現在テキスト要素の長さのために取得される（１６４６）。次に、スイッチ・オペレーション１６４８がフォーマットに基づいて実行される。このインプリメンテーションで利用できるフォーマットはリスト、セグメント配列、範囲およびチェーンである。フォーマットがリスト・フォーマットであれば、リスト・フォーマット処理１６５０が実行される。フォーマットがセグメント配列であれば、セグメント配列フォーマット処理１６５２が実行される。フォーマットが範囲であれば、範囲フォーマット処理１６５４が実行される。フォーマットがチェーンであれば、チェーン・フォーマット処理１６５６が実行される。ブロック１６５０－１６５６に続いて、結果が戻され（１６５８）、これによりテキスト要素ルックアップ処理１６３０は完了する。

【０１６７】

本発明の第１側面によれば、図２０はチェーン・フォーマット処理１６６０を示すフローチャートである。チェーン・フォーマット処理１６６０は図１９に示すチェーン・フォーマット処理１６３５によって実行される処理である。

【０１６８】

チェーン・フォーマット処理１６６０はチェーン内のテーブル数のチェーン・カウントを取得する（１６６２）。そのあと、現在のカウントはゼロにセットされる（１６６４）。次に、現在のカウントがチェーン・カウントより大であるか等しいかどうかの判断１６６６が行われる。現在のカウントがチェーン・カウントより大または等しければ、チェーン・フォーマット処理１６６０は戻り（１６６８）、結果が見つからなかったのでエラーを通知する。他方、現在のカウントがチェーン・カウントより大でも等しくもなければ、現在のルックアップ・テーブルとそのフォーマットが取得される（１６７０）。このインプリメンテーションで使用されるフォーマットは図１９で使用されたものと同じである。そのあとスイッチ・オペレーション１６７２がフォーマットに基づいて実行される。フォーマットがリストであれば、リスト・フォーマット処理１６７４が実行される。フォーマットがセグメント配列であれば、セグメント配列フォーマット処理１６７６が実行される。フォーマットが範囲であれば、範囲フォーマット処理１６８０が実行される。フォーマットがチェーンであれば、チェーン・フォーマット処理１６８２が実行される。ブロック１６１４～１６２２に続いて、現在のカウントがインクリメントされる（１６８４）。次に、結果が見つかったかどうかの判断１６８６が行われる。結果が見つからなければ、チェーン・フォーマット処理１６６０はループして判断ブロック１６６６に戻り、テーブル・チェーン内の次のルックアップ・テーブルを調べていく。しかるに、結果が見つかったと判断１６８６されたときは、結果が戻され（１６８８）、これによりチェーン・フォーマット処理１６６０は完了する。

【０１６９】

本発明の第１側面によれば、図２１は範囲フォーマット処理１７００を示すフローチャートである。範囲フォーマット処理１７００は図１９のブロック１６５４と図２０のブロック１６７８によって実行される処理である。範囲フォーマットはデルタ値が各フィールドと関連づけられている文字の範囲リストである。

【０１７０】

範囲フォーマット処理１７００はそのサブテーブルのサブセット・フラグが選択フラグに一致しているかどうかを判断（１７０２）することから始まる。選択フラグは選択マスクに対するもので、サブセット・フラグはサブテーブル・マスクに対するものである。一致していなければ、範囲フォーマット処理は戻り（１７０４）、結果が見つからなかったことをエラー・コードで通知する。他方、そのサブテーブルのサブセット・フラグが選択フラグに一致していることを判断１７０２が示していれば、テキスト要素の長さが１より大であるかどうかの判断１７０６が行われる。テキスト要素の長さが１より大であれば、このフォーマットは誤って選択されたことになる。この特定インプリメンテーションでのマッピング・テーブルの構成は範囲フォーマットが長さ１のテキスト要素だけを目的とするようになっているためである。従って、テキスト要素が１より大であれば、ブロック１

７０４が実行されて戻り、結果が見つからなかったことを通知する。他方、テキスト要素の長さが１より大でなければ、範囲フォーマット処理１７００は続行される。範囲フォーマットをもつサブテーブルは範囲配列をもち、各範囲はそれぞれに関連づけられたデルタ値をもっている。次に、範囲配列がサーチされ（１７０８）、変換されるユニコード文字の該当する範囲が見つけられる。範囲が見つかったかどうかの判断１７１０が行われる。範囲が見つからなければ、範囲フォーマット処理１７００は戻り（１７０４）、結果が見つからなかったことをエラー・コードで通知する。しかるに、結果が見つかったときは、その範囲の対応するデルタ値が取得される（１７１２）。次に、このデルタ値はユニコード値に加えられる（１７１４）。そのあと、その結果は出力シーケンスにマッピングされる（１７１６）。結果を出力シーケンスにマッピングする処理は図２２Ａ～図２２Ｃを参照して以下で詳しく説明する。ブロック１７１６に続いて、範囲フォーマット処理は完了し、戻る。

【０１７１】

本発明の第１側面によれば、図２２はリスト・フォーマット処理１８００を示すフローチャートである。リスト・フォーマット処理１８００は図１９のブロック１６５０と図２０のブロック１６７４によって実行される処理である。リスト・フォーマットはテキスト要素が配列されたリストであり、この配列リストまでのインデックスｉは対応するルックアップ・ターゲット・リストまでのインデックスである。

【０１７２】

リスト・フォーマット処理１８００はそのサブテーブルのサブセット・フラグが選択フラグに一致しているかどうかの判断１８０２から始まる。一致していなければ、リスト・フォーマット処理１８００は戻り（１８０４）、マッピングが見つからなかったことをエラー・コードで通知する。他方、そのサブテーブルのサブセット・フラグが選択フラグに一致していれば、リスト内のテキスト要素に基づいて最適化バイナリ・サーチが行われる（１８０６）。次に、サーチがリスト内のテキスト要素を見つけたかどうかの判断１８０８が行われる。見つからなければ、再びリスト処理は戻り（１８０４）、マッピングが見つからなかったことを通知する。しかし、テキスト要素が見つかれば、テキスト要素が見つかった個所のインデックスｉが取得される（１８１０）。このインデックスｉはルックアップ・ターゲットを取得する（１８１２）ために使用される。次に、ルックアップ・ターゲットは出力シーケンスにマッピングされる（１８１４）。これでリスト・フォーマット処理は完了し、戻る。

【０１７３】

本発明の第１側面によれば、図２３Ａと図２３Ｂはセグメント配列フォーマット処理１９００を示している。セグメント配列フォーマット処理１９００は図１９のブロック１６５２と図２０のブロック１６７６によって実行される処理である。セグメント配列フォーマットは第１テキスト要素配列、最終テキスト要素配列、およびｎ個のオフセット配列を含んでいる。オフセット配列内のオフセットは種々のルックアップ・ターゲット・リストを指している。

【０１７４】

セグメント配列フォーマット処理１９００はそのサブテーブルのサブセット・フラグが選択フラグに一致しているかどうかの判断１９０１から始まる。サブセット・フラグが選択フラグに一致していれば、最適化サーチが行われる（１９０２）。最適化サーチは最終テキスト要素配列内にあって、探索していたテキスト要素よりも大であるか等しい最小エントリを見つけ、そのエントリのインデックスｉを取得する。次に、第１テキスト要素配列内のｉ番目のエントリが探索していたテキスト要素より小であるか等しいかどうかの判断１９０４が行われる。そうでなければ、セグメント配列フォーマット処理１９００は戻り（１９０６）、マッピングが見つからなかったことをエラー・コードで知らせる。また、判断１９０１が失敗した場合には、ブロック１９０６も実行される。他方、最初のテキスト要素配列内のｉ番目のエントリが探索していたテキスト要素より小であるか等しいことを判断１９０４が示しているときは、ｉ番目のエントリは探索していたテキスト要素に

対応することが分かる。そのあと、ルックアップ・ターゲット・リストがオフセット配列内のｉ番目のエントリを通して取得される（１９０８）。すなわち、オフセット配列内のｉ番目のエントリに入っているオフセットはルックアップ・ターゲット・リストを示している。ルックアップ・ターゲット・リストまでのインデックスｊが次に判断される。インデックスｊは探索していたテキスト要素の値から、取得（１９０８）したルックアップ・ターゲット・リスト（または範囲）内の第１テキスト要素を引いた値が与えられる。インデックスｊをもつルックアップ・ターゲットが次にルックアップ・ターゲット・リストから取得される（１９１２）。次に、ルックアップ・ターゲットがゼロに等しいかどうかの判断１９１４が行われる。ルックアップ・ターゲットがゼロに等しければ、セグメント配列フォーマット処理は戻り、マッピングが見つからなかったことをエラー・コードで通知する。他方、ルックアップ・ターゲットがゼロに等しくなければ、ルックアップ・ターゲットは出力シーケンスにマッピングされる（１９１８）。ブロック１９１８に続いて、セグメント配列フォーマット処理１９００は完了し、戻る。

【０１７５】

図２４はサーチ変種リスト処理２０００を示すフローチャートである。サーチ変種リスト処理２０００は図１９のブロック１６３２によって実行される処理である。言い換えれば、サーチ変種リスト処理２０００はマッピング・テーブル４１４を使用してルックアップ・ハンドラ４１２によって実行されるルックアップ・テキスト要素処理１６３０の一部である。

【０１７６】

サーチ変種リスト処理２０００は変種リスト内の要素の総カウントを取得する（２００２）。そのあと、現在のカウントがゼロに初期化される（２００４）。現在のカウントが総カウントより大であるか等しいかどうかの判断２００６が行われる。現在のカウントが総カウントより大または等しければ、サーチ変種リスト処理２０００は戻り（２００８）、変種が見つからなかったことをエラー・コードで通知する。他方、現在のカウントが総カウントより大または等しくなければ、変種リスト内にあって、現在のカウントに関連するエントリが実際の属性および要求された変種に一致しているかどうかの判断２０１０が行われる。一致していれば、変種リスト内のエントリからの変種フラグが戻される（２０１２）。一致していなければ、現在のカウントはインクリメントされ（２０１４）、そのあと処理はブロック２００６に戻り、変種の１つが一致するか、あるいは変種のすべてが考慮されるまでループを続けて変種リスト内の使用可能な変種を見つける。

【０１７７】

図２５Ａおよび図２５Ｂは変種リスト２１００を示す概略図である。図２５Ａに示すように、変種リスト２１００は変種領域２１０２、所望属性領域２１０４および変種フラグ領域２１０６を含んでいる。図２５Ｂは好ましいインプリメンテーションによる実際属性ビット・マスクを示す図である。実際属性ビット・マスク２１０８は３２ビット変数であり、シメトリック・スワッピング状態を示す第１部分２１１０（ビット０と１）、垂直または水平形式を示す第２部分（ビット２と３）、解明方向を示す第３部分（ビット８と９）、およびコンテキストを示す第４部分（ビット１６～１９）をもっている。本発明の第３側面によれば、第４部分２１１６のビットはコンテキスト処理９８４（図１０Ｃ）で判断されたコンテキスト・マスクによってセットされる。さらに、本発明の第１ないし第４側面によれば、部分内の各ビットはフラグを表している。ビットはフラグが未知であるか、あるいはセットされていなければ、値が“０”であり、セットされているときは値は“１”になっている。呼び出し側は第２部分２１１０をセットし、スキャナ４０８は第１、第２および第４部分２１１０、２１１４および２１１６をセットする。

【０１７８】

所望属性ビット・マスクは実際属性ビット・マスクと同じフォーマットであるが、ビットは属性のどれが特定テーブルおよび変種の正しいマッピングを得るために重要であるかに応じてセットされる（これはマッピング・テーブル４１４の設計によって決まる）。所望属性ビット・マスク内のビットはマッピング判断の際に考慮される各属性には“１”が

セットされ、ある部分のすべてのビットが“１”にセットされていれば、属性はマッピング期間には無視される。例えば、ビット０が“１”で、ビット１が“０”ならば、シメトリック・スワッピングはオンであり、マッピングが行われるとき考慮される。他方、ビット０と１が共に“１”であれば、シメトリック・スワッピングは完全に無視される。所望属性ビット・マスクの残りの未使用ビットは“１”にセットされ、必要ならばあとで値を割り当てることができる。以下、所望属性ビット・マスクの例をいくつか示して説明する。

【０１７９】

方向が左から右へであり、他の属性はいずれも重要でないとする。そうすると所望属性ビット・マスクはｘＦＦＦＦＦＤＦＦとなる。これに対して、方向が右から左へであり、シメトリック・スワッピングがオンであるとすると、所望属性ビット・マスクはｘＦＦＦＦＦＥＦＤとなる。方向が右から左へで、シメトリック・スワッピングがオフであれば、所望属性ビット・マスクはｘＦＦＦＦＦＥＦＥとなる。上記の異種所望属性ビット・マスクの各々によると、異なる変換コードを選択することができる。例えば、ユニコード文字ｕ００２８をＭａｃＡｒａｂｉｃにマッピングすると、所望属性ビット・マスクｘＦＦＦＦＦＤＦＦではｘ２８が、所望属性ビット・マスクｘＦＦＦＦＦＥＦＤではｘＡ８が、所望属性ビット・マスクｘＦＦＦＦＦＥＦＥではｘＡ９が得られる。

【０１８０】

本発明の第１側面によれば、図２６Ａ、図２６Ｂおよび図２６Ｃはルックアップ・ターゲットの出力シーケンスへのマッピング処理２２００を示すフローチャートである。ルックアップ・ターゲットの出力シーケンスへのマッピング処理２２００は図２２のブロック１８１４および図２３Ａのブロック１９１８に関連している。

【０１８１】

ルックアップ・ターゲットの出力シーケンスへのマッピング処理２２００は間接が許されるかどうかの判断２２０２から始まる。間接が許されなければ、ルックアップ・ターゲットはｃｈａｒｓｉｚｅの断片で出力シーケンスにコピーされる（２２０４）。ｃｈａｒｓｉｚｅとは、ターゲット・コード化内の文字の最小サイズのことであり、マッピング・テーブル４１４のヘッダ５００に指定されている。ブロック２２０４に続いて、処理２００は完了し、結果が見つかったとの通知を出して戻る（２２０６）。他方、間接が許されると判断２２０２が判断したときは、ルップアップ・ターゲットの上位ビットが１に等しいかどうかの判断２２０８が行われる。上位ビットが１に等しくなければ、第１バイトがヌル出力シーケンス（つまり、長さが０の出力シーケンス）を示しているかどうかの判断２２１０が行われる。第１バイトがｘ７Ｆを示していれば、ヌル出力シーケンスが示される（２２１２）（一致するものが見つかったが、文字は出力シーケンスに追加されない）。他方、第１バイトがヌル出力シーケンスを示していないときは、ｃｈａｒｓｉｚｅの断片が出力シーケンスにコピーされる（２２１４）。その場合には、第１バイトは出力シーケンスの長さを示していることが好ましい。ブロック２２１２と２２１４に続いて、ブロック２２０６が実行され、これにより処理は完了し、結果が見つかったことを通知して戻る。

【０１８２】

しかるに、ルックアップ・ターゲットの上位ビットが１に等しいと判断２２０８が判断した場合には、ルックアップ・ターゲットは所望の出力シーケンスを間接的に参照しているので追加の処理が必要になる。具体的には、間接シーケンスまでのオフセットがルックアップ・ターゲットの残余部分を使用して指定される（２２１６）。ルックアップ・ターゲットの残余部分の上位ビットが１に等しいかどうかの判断２２１８が行われる。等しくなければ、ｃｈａｒｓｉｚｅの断片が出力シーケンスにコピーされ（２２２０）、そのあと処理２２００はマッピングが見つかったとの通知を戻して（２２００）完了する。他方、ルックアップ・ターゲットの残余部分の上位ビットが１に等しいと判断されていれば（２２１８）、順次チェーン内のシーケンスのカウントが取得され（２２２４）、リニア・サーチがシーケンスを通るように行われ、マッピングが特定される。ブロック２２２６に

続いて、前述したブロック２２２０と２２２２が実行される。

【０１８３】

図２６Ｃは図２６Ｂのブロック２２２６によって実行されるオペレーションの詳細図である。すでに説明したように、図２６Ｃのブロック２２２６はシーケンスを通るリニア・サーチを実行する。最初に、現在のカウントがゼロにセットされる（２２２８）。現在のカウントがシーケンス・カウントより大または等しいかどうかの判断２２２８が行われる。現在のカウントがシーケンス・カウントより大または等しければ、ルックアップ・ターゲットの出力シーケンスへのマッピング処理２２００はマッピングが見つからなかったとの通知と共に戻る（２２３２）。他方、現在のカウントがシーケンス・カウントより大または等しくなければ、次の出力シーケンスとそのシーケンス・マスクが取得される（２２３４）。シーケンス・マスクは複数の出力シーケンスの１つを選択するために使用されるマスクである。次に、サブセット・フラグが使用中であるかどうかの判断２２３６が行われる。サブセット・フラグが使用中でなければ、シーケンス・マスクと論理ＡＮＤがとられた実際属性が実際属性にビットワイズに等しいかどうかの判断２２３８が行われる。そうであれば、マッピングが見つかったので処理は上述したブロック２２２０と２２２２を実行する。他方、サブセット・フラグが使用中であると判断（２２３６）されたときは、シーケンス・マスクと論理ＡＮＤがとられた選択フラグがビットワイズにシーケンス・マスクと等しいかどうかの判断２２４０が行われる。そうであれば、マッピングが見つかったので前述したブロック２２２０と２２２２が実行される。従って、ブロック２２３８と２２４０は異なった方法で正しい出力シーケンスを取得する。他方、どちらかの判断ブロック２２３８または２２４０が現在のシーケンスが正しいマッピングでないと示していれば、現在のカウントはインクリメントされ（２２４２）、処理はブロック２２３０に戻ってチェーン内の次のシーケンスのオペレーションを続ける。

【０１８４】

図２７は本発明によるフォールバック・ハンドリング処理２３００を示すフローチャートである。フォールバック・ハンドリング処理２３００はフォールバック・ハンドラ４１６によって実行され、図７に示すユニコード・コンバータ処理７００のブロック７１６によって起動される処理である。

【０１８５】

フォールバック・ハンドリング処理２３００はフォールバック・オプションを使用してテキスト要素を調べる（２３０２）。ルックアップ２３０２は図１９を参照して上述したテキスト要素ルックアップ処理１６３０に類似している。唯一の重要な違いは、変化した選択フラグを通して追加サブセットが考慮されるようにフォールバック・オプションがセットされていることである。次に、テキスト要素の変換コードが見つかったかどうかの判断２３０４が行われる。変換コードが見つかっていれば、変換またはマッピングを取得するときにどのフォールバックが使用されたかを示すようにエラー・コードがセットされる（２３０６）。ブロック２３０６に続いて、フォールバック・ハンドリング処理２３００は完了し、戻る。

【０１８６】

他方、判断２３０４が変換コードが見つからなかったと示しているときは、フォールバック・オプションに基づいてスイッチ・オペレーション２３０８が実行される。フォールバック・オプションには、次のものがある。すなわち、デフォルト、呼び出し側定義、呼び出し側定義を伴うデフォルト、またはデフォルトを伴う呼び出し側定義である。フォールバック・オプションがデフォルトであれば、スイッチ・オペレーション２３０８はデフォルト処理２３１０を実行させる。フォールバック・オプションが呼び出し側定義であれば、スイッチ・オペレーション２３０８は呼び出し側定義のオペレーション２３１２を実行させる。フォールバック・オプションが呼び出し側定義を伴うデフォルトであれば、スイッチ・オペレーション２３０８はデフォルト処理を実行させ（２３１４）、続いて判断２３１６と呼び出し側定義の処理を実行させる（２３１８）。判断２３１６はデフォルト処理が正常に行われると、呼び出し側定義処理２３１８をバイパスするように働く。フォ

ールバック・オプションがデフォルト処理を伴う呼び出し側定義であれば、スイッチ・オペレーション２３０８は呼び出し側定義処理を実行させ（２３２０）、続いて判断２３２２とデフォルト処理を実行させる（２３２４）。判断２３２２は、呼び出し側定義処理２３２０が正常に行われたときは、デフォルト処理２３２４をバイパスするように働く。スイッチ・オペレーション２３０８に関連する処理に続いて、フォールバック処理２３００がマッピングまたは変換コードを正常に特定していたかどうかの判断２３２６が行われる。フォールバック処理２３００が失敗していれば、文字セットのデフォルト・フォールバック文字シーケンスが取得される（２３２８）。デフォルト・フォールバック文字シーケンスとは、フォールバック・ルックアップ２３０２が変換コードを特定するのに失敗したとき使用される変換コードである。好ましくは、デフォルト・フォールバック文字シーケンスはマッピング・テーブル４１４のヘッダに収められている。例えば、ＡＳＣＩＩでは、デフォルト・フォールバック文字シーケンスは“？”であるのが普通である。そのあと、ブロック２３２８に続いて、またはフォールバック処理がマッピングまたは変換コードを取得するのに成功したときはブロック２３２６に続いて、フォールバック・オプションが使用されたことを示すエラー・コードがセットされ（２３０６）、フォールバック・ハンドリング処理２３００は完了し、戻る。

【０１８７】

図２８はデフォルト処理２４００を示すフローチャートである。デフォルト処理は図２７のブロック２３１０、２３１４および２３２４で実行される処理と関連している。

【０１８８】

デフォルト処理２４００は最初に現在のカウントをゼロにセットする（２４０２）。次に、現在のカウントがテキスト要素の長さより大であるか等しいかの判断２４０４が行われる。そうであれば、デフォルト処理２４００は完了し、戻る。そうでなければ、フォールバック・フラグがセットされた単一ユニコード文字に対してルックアップ処理が実行される（２４０６）。ここでは、ルックアップはテキスト要素の個別文字に対するものであるが、以前（ブロック３２０２）では、ルックアップはテキスト要素全体に対するものであった。そのあと、単一ユニコード文字の変換コードが見つかったかどうかの判断２４０８が行われる。見つからなければ、ユニコード文字で使用できる個別マッピングがなかったことをエラー・コードで通知してデフォルト処理２４００は戻る（２４１０）。他方、変換コードが見つかっていれば、現在のカウントがインクリメントされ（２４１２）、処理はテキスト要素内の次のユニコード文字の処理を行うためにブロック２４０４に戻る。

【０１８９】

上述した図４～図２８はユニコードからターゲット・コード化への変換（ユニコードから）に関するものであるが、上述したように、ユニコード・コード変換システム３００は異なるソース・コード化からユニコードに変換する（ユニコードへ）機能も等しく備えている。ユニコードへはユニコードからの処理と類似しているが、実質的には複雑度が軽減されている。ユニコードへの処理はターゲット・コード化を判断するときテキスト要素を探すためにスキャンしたり、複数の文字シーケンスを調べたりする必要がないのが通常である。ユニコードへの処理はソース文字列を個々の文字に分割し、そのあとでユニコードの対応するコード・ポイントを見つけるだけである。しかし、文字のマッピングがそのあとに続く文字に影響されるようなまれな場合には（例えば、デーヴァナーガリーのようなインド・スクリプト）、上述したようにスキャンが行われる場合がある。

【０１９０】

図２９は本発明によるユニコード・コード変換システム２５００の実施例を示すブロック図である。ユニコード・コード変換システム２５００はユニコードへの変換を行う（つまり、ユニコードへの処理）。ユニコード・コード変換システム２５００はユニコードへのコンバータ２５０２を含み、このコンバータはソース文字列２５０４を受信し、ユニコード文字列２５０６を出力する。ユニコードへのコンバータ２５０２はスキャナ２５０８とやりとりするユニコードへのコンバータを通してコード変換プロセスを実行する。スキ

ャナ２５０８はスキャナ・テーブル２５１０を使用して、ソース文字列２５０４をスキャンしソース文字列２５０４を文字に分断する。ここでは、ユニコードからの場合と異なり、ソース文字列は個々の文字に分割されるだけである。次に、ユニコードへのコンバータ２５０２はルックアップ・ハンドラ２５１２を使用して個別文字を調べそのユニコード・コード化を取得する。ルックアップ・ハンドラ２５１２はマッピング・テーブル２５１４を使用してユニコードの文字を取得する。さらに、ユニコードへのコンバータ２５０２はフォールバック・ハンドラ２５１６を使用することもできる。フォールバック・ハンドラ２５１６はマッピング・テーブル２５１４と一緒に動作して、ルックアップ・ハンドラ２５１２がユニコード文字を特定できなかった場合に、テキスト要素のフォールバック・マッピングとして使用できるターゲット・コード化内の１つまたは複数の文字を特定する。

【０１９１】

スキャナ２５０８、スキャナ・テーブル２５１０、ルックアップ・テーブル２５１２、マッピング・テーブル２５１４、フォールバック・ハンドラ２５１６、状態管理機構２５１８は図４の対応するデバイスと類似しているが、実質的には複雑度が軽減されている。従って、これらのデバイスはユニコードへとユニコードからの両方の処理を実現するように設計することができる。さらに、与えられたコンピュータ・システムや他のエレクトロニック・デバイスがユニコードへとユニコードからの両方の処理を実行する機能を備えているときは、そのコンピュータ・システムや他のエレクトロニック・デバイスはハブとして動作することができる。このハブはユニコードがサポートする種々の各国語文字間で変換を行うように動作することができる。例えば、ソースの各国語文字セットはまずユニコードに変換され、次にターゲットの各国語文字セットに変換される。

【０１９２】

本発明の第１側面によれば、上述したテーブルは、好ましくは、上述したフォーマットを使用してテーブルにストアすべきデータの圧縮を行う。さらに、本発明の第１側面によれば、圧縮を重視している。例えは、圧縮すると、属性テーブルは５０倍（５０対１）に小さくなる。しかし、本発明の第１側面によれば、この分野の精通者ならば理解されるように、テーブルのフォーマットは多数の形体をとることができるので、あるテーブルは他のテーブルよりも実現が容易になる。最も容易な実現はどの圧縮も使用しないテーブルであるが、そのためには最大のデータ記憶装置が必要になる。さらに、本発明の第１側面によれば、テーブルは上述したインプリメンテーションで使用されているが、これらのテーブルで引き起こされる振舞を直接にコーディングすることも可能である。しかし、直接コーディングはコード変換システムのオペレーションの変更を困難にするのに対し、テーブルを使用すると、そのような変更を取り入れるためにテーブルだけを変更するだけで済むのが普通である。

【０１９３】

本発明の多数の特徴および利点は上述してきた説明で明らかにした通りであり、本発明のかかる特徴と利点はすべて請求の範囲に記載されている。さらに、本発明はこの分野の精通者が容易に理解されるように、種々態様に変更することが可能であるので、本発明は上述してきた正確な構造とオペレーションに限定されるものではない。従って、すべての適当な変更および等価技術は本発明の範囲に属するものである。

【図面の簡単な説明】

【０１９４】

【図１】本発明による代表的なコンピュータ・システムを示すブロック図である。

【図２】ユニコード文字コード化のフォーマットを示す図である。

【図３】ソース文字列を受信し、ターゲット文字列を出力する本発明による基本的ユニコード・コード変換システムを示すブロック図である。

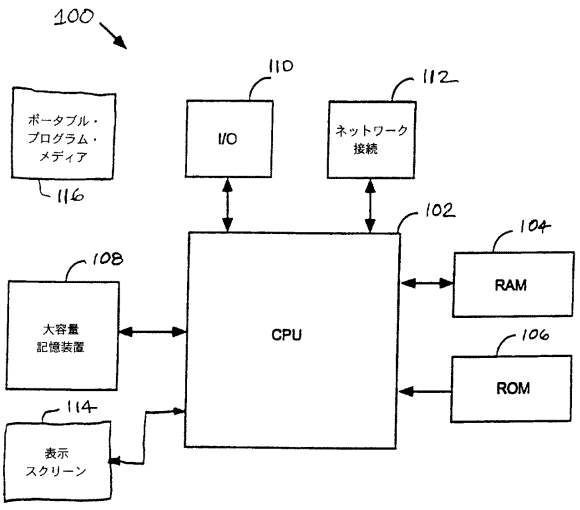
【図４】本発明の実施例によるユニコードからのコード変換システムの実施例を示すブロック図である。

【図５】ユニコード・コード変換システムのマッピング・テーブルの好ましい配列を示す概略図である。

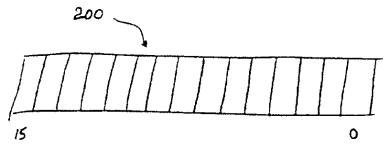
【図６Ａ】本発明の実施例によるユニコード・コード変換システムを利用するアプリケーション・プログラムによって実行される処理を示すフローチャートである。
【図６Ｂ】本発明の実施例による切り捨て処理を示すフローチャートである。
【図７】本発明の実施例によるユニコード・コンバータ処理を示すフローチャートである。
【図８】本発明の実施例による更新オフセット処理を示すフローチャートである。
【図９Ａ】本発明の実施例による次のテキスト要素処理を示すフローチャートである。
【図９Ｂ】本発明の実施例による次のテキスト要素処理を示すフローチャートである。
【図１０Ａ】本発明の実施例による次のテキスト要素処理を示すフローチャートである。
【図１０Ｂ】本発明の実施例による次のテキスト要素処理を示すフローチャートである。
【図１０Ｃ】本発明の実施例による次のテキスト要素処理を示すフローチャートである。
【図１１】本発明の実施例によるスキャナを示すブロック図である。
【図１２】図１０に示す属性テーブルの好ましいフォーマットを示す概略図である。
【図１３】本発明の実施例による属性ルックアップ処理を示すフローチャートである。
【図１４Ａ】本発明の好適実施例による次のアクションを判断するためにスキャナによって利用されるスキャナ・テーブルを示す概略図である。
【図１４Ｂ】本発明の好適実施例による次のアクションを判断するためにスキャナによって利用されるスキャナ・テーブルを示す概略図である。
【図１５】本発明の実施例による好ましいレイアウトと、スキャナ・テーブルにストアされる情報の両方を表しているテーブルである。
【図１６Ａ】本発明の好適実施例による好ましいレイアウトと、スキャナ・テーブルにストアされる情報の両方を表しているテーブルである。
【図１６Ｂ】本発明の好適実施例による好ましいレイアウトと、スキャナ・テーブルにストアされる情報の両方を表しているテーブルである。
【図１７Ａ】本発明の実施例による方向解明処理を示すフローチャートである。
【図１７Ｂ】本発明による双方向状態テーブルの好ましいレイアウトを表しているテーブルである。
【図１７Ｄ】本発明による双方向状態テーブルの好ましいレイアウトを表しているテーブルである。
【図１８】本発明の実施例による初期方向処理を判断するためのフローチャートである。
【図１９】本発明の実施例によるルックアップ・テキスト要素処理を示すフローチャートである。
【図２０】本発明の実施例によるチェーン・フォーマット処理を示すフローチャートである。
【図２１】本発明の実施例による範囲フォーマット処理を示すフローチャートである。
【図２２】本発明の実施例によるリスト・フォーマット処理を示すフローチャートである。
【図２３Ａ】本発明の実施例によるセグメント配列フォーマット処理を示す図である。
【図２３Ｂ】本発明の実施例によるセグメント配列フォーマット処理を示す図である。
【図２４】本発明の実施例による変形リスト・サーチ処理を示すフローチャートである。
【図２５Ａ】本発明の実施例による変形リスト処理に関連するテーブルとデータ・フォーマットを示す概略図である。
【図２５Ｂ】本発明の実施例による変形リスト処理に関連するテーブルとデータ・フォーマットを示す概略図である。
【図２６Ａ】本発明の実施例による出力シーケンスへのルックアップ・ターゲットのマッピング処理を示すフローチャートである。
【図２６Ｂ】本発明の実施例による出力シーケンスへのルックアップ・ターゲットのマッピング処理を示すフローチャートである。
【図２６Ｃ】本発明の実施例による出力シーケンスへのルックアップ・ターゲットのマッピング処理を示すフローチャートである。

【図２７】本発明の実施例による本発明の処理に従うフォールバック・ハンドリング処理を示すフローチャートである。
【図２８】本発明の実施例によるデフォルト処理を示すフローチャートである。
【図２９】本発明の実施例によるユニコードへのコード変換の実施例を示すブロック図である。
【符号の説明】
【０１９５】
１００　コンピュータ・システム
２００　フォーマット
４００　ユニコード・コード変換システム
４１４　マッピング・テーブル
１００４　属性テーブル
１３００　スキャナ・テーブル
１３０２　要素
１４００　テーブル
１５１１　双方向状態テーブル
２１００　変種リスト
２１０８　実際属性ビット・マスク
２１１０，２１１２，２１１４，２１１６　部分

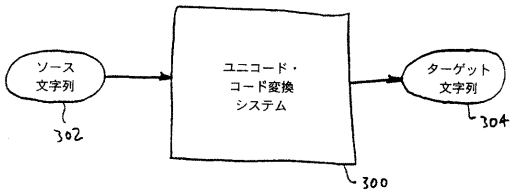
【図１】



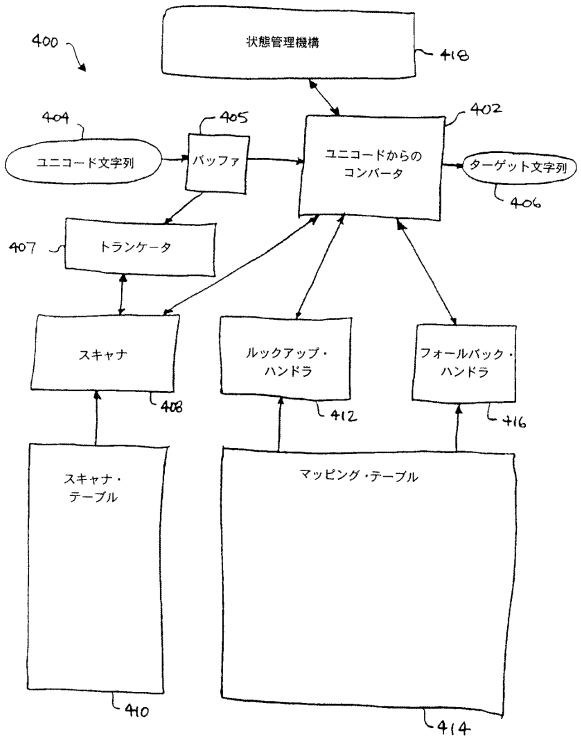
【図２】



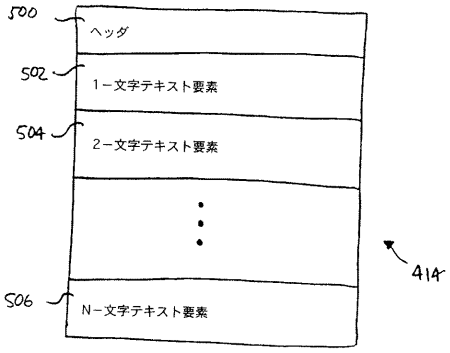
【図３】



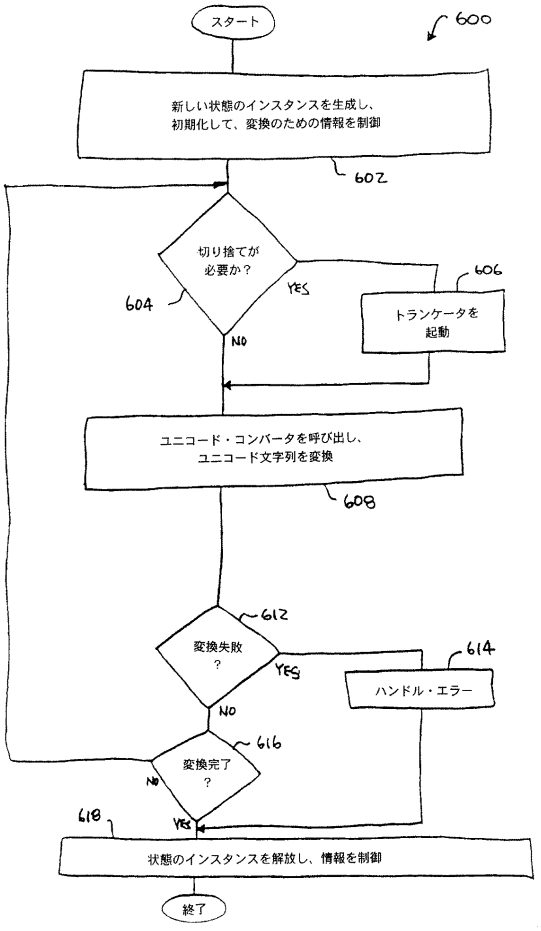
【図４】


400
状態管理機構
418
404
405
402
ユニコード文字列
バッファ
ユニコードからの
コンバータ
ターゲット文字列
406
407
トランケータ
スキャナ
ルックアップ・
ハンドラ
フォールバック・
ハンドラ
408
412
416
スキャナ・
テーブル
マッピング・テーブル
410
414

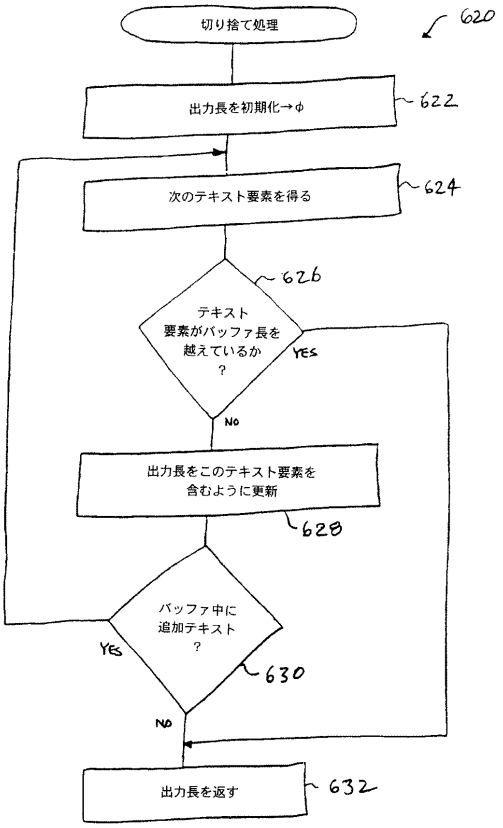
【図５】


500
ヘッダ
502
1－文字テキスト要素
504
2－文字テキスト要素
506
N－文字テキスト要素
414

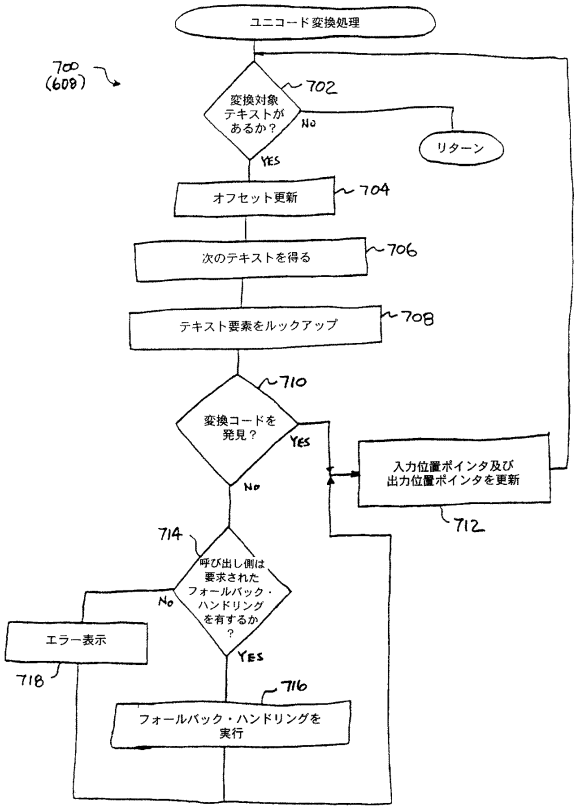
【図６Ａ】


スタート
600
新しい状態のインスタンスを生成し、
初期化して、変換のための情報を制御
602
切り捨てが
必要か？
604
YES
606
トランケータを
起動
NO
ユニコード・コンバータを呼び出し、
ユニコード文字列を変換
608
612
変換失敗
？
YES
614
ハンドル・エラー
NO
616
変換完了
？
NO
YES
618
状態のインスタンスを解放し、情報を制御
終了

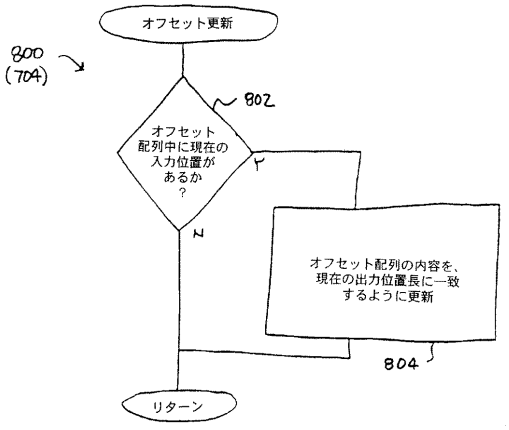
【図６Ｂ】


切り捨て処理
620
出力長を初期化→φ
622
次のテキスト要素を得る
624
626
テキスト
要素がバッファ長を
越えているか
？
YES
NO
出力長をこのテキスト要素を
含むように更新
628
バッファ中に
追加テキスト
？
YES
630
NO
出力長を返す
632

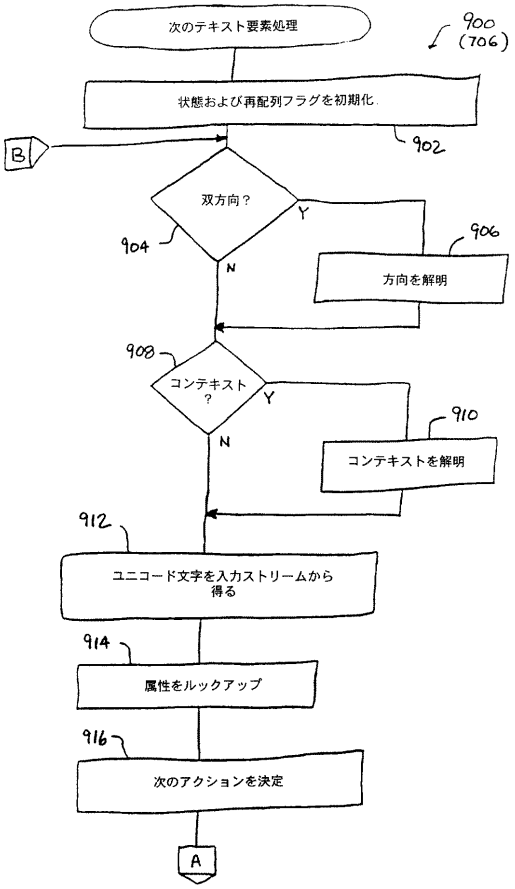
【図７】


ユニコード変換処理
700
(608)
変換対象テキストがあるか？
702
No
リターン
YES
オフセット更新
704
次のテキストを得る
706
テキスト要素をルックアップ
708
変換コードを発見？
710
Yes
No
入力位置ポインタ及び出力位置ポインタを更新
712
呼び出し側は要求されたフォールバック・ハンドリングを有するか？
714
No
エラー表示
718
Yes
フォールバック・ハンドリングを実行
716

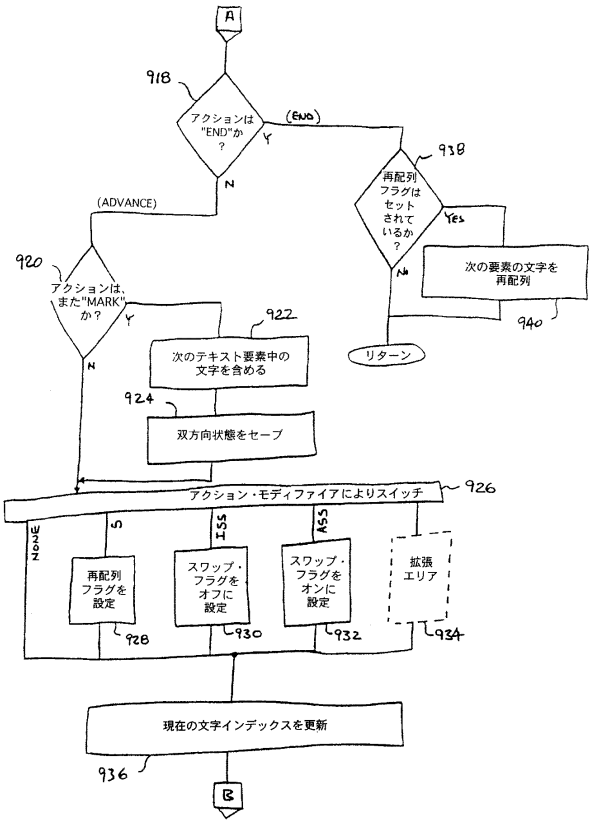
【図８】


オフセット更新
800
(704)
オフセット配列中に現在の入力位置があるか？
802
Y
N
オフセット配列の内容を、現在の出力位置長に一致するように更新
804
リターン

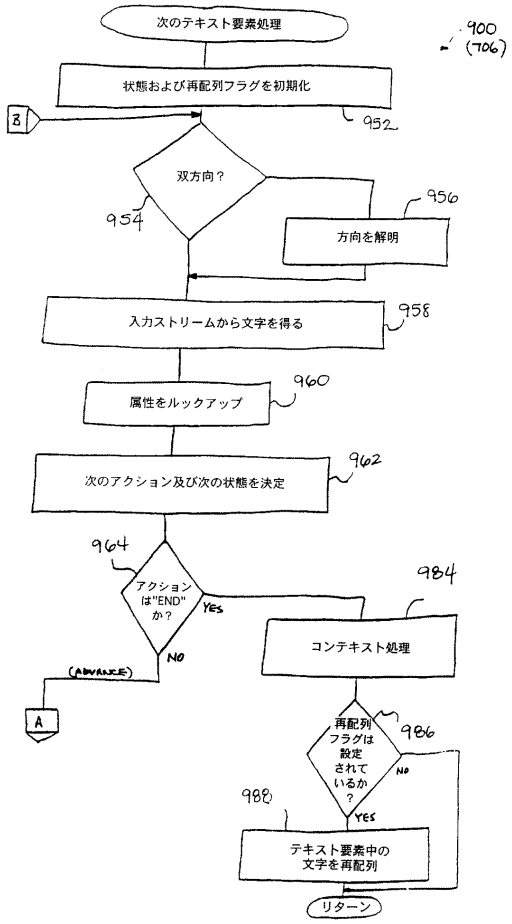
【図９Ａ】


次のテキスト要素処理
900
(706)
状態および再配列フラグを初期化
902
B
双方向？
904
Y
N
方向を解明
906
コンテキスト？
908
Y
N
コンテキストを解明
910
ユニコード文字を入力ストリームから得る
912
属性をルックアップ
914
次のアクションを決定
916
A

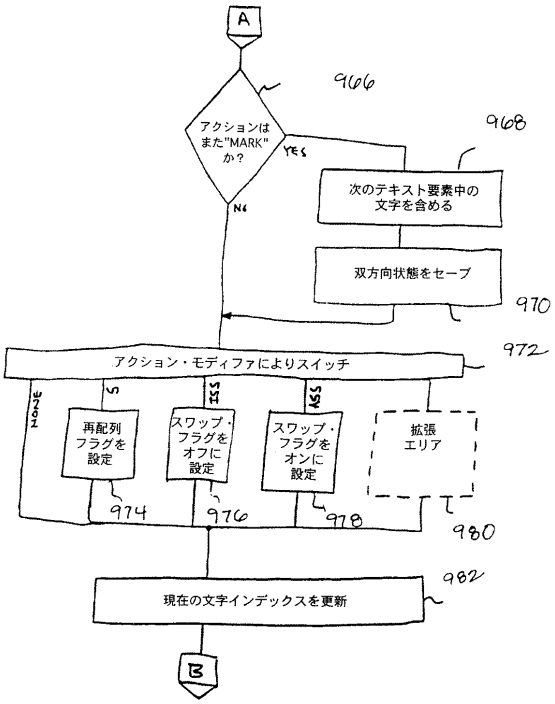
【図９Ｂ】


A
アクションは"END"か？
918
(END)
Y
N
再配列フラグはセットされているか？
938
Yes
No
次の要素の文字を再配列
940
リターン
(ADVANCE)
アクションは、また"MARK"か？
920
Y
N
次のテキスト要素中の文字を含める
922
双方向状態をセーブ
924
アクション・モディファイアによりスイッチ
926
NONE
再配列フラグを設定
928
SET
スワップ・フラグをオフに設定
930
ASS
スワップ・フラグをオンに設定
932
拡張エリア
934
現在の文字インデックスを更新
936
B

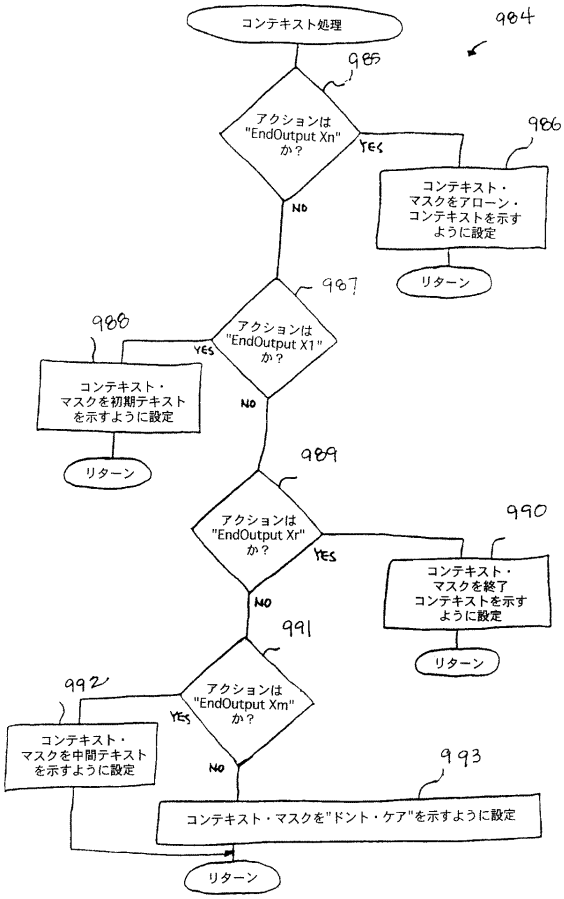
【図１０Ａ】



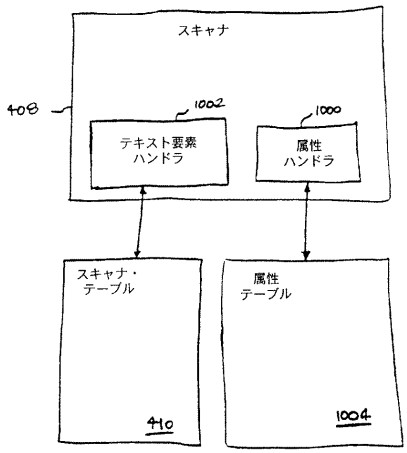
【図１０Ｂ】



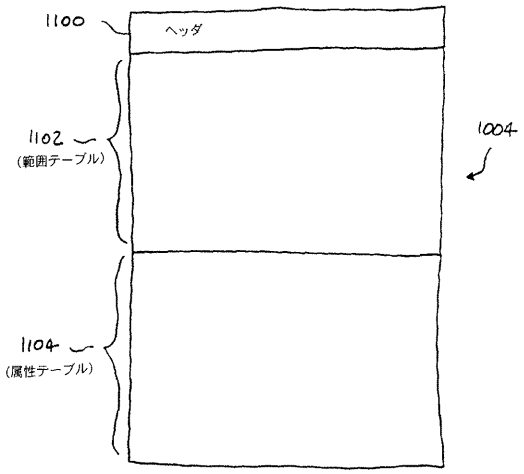
【図１０Ｃ】



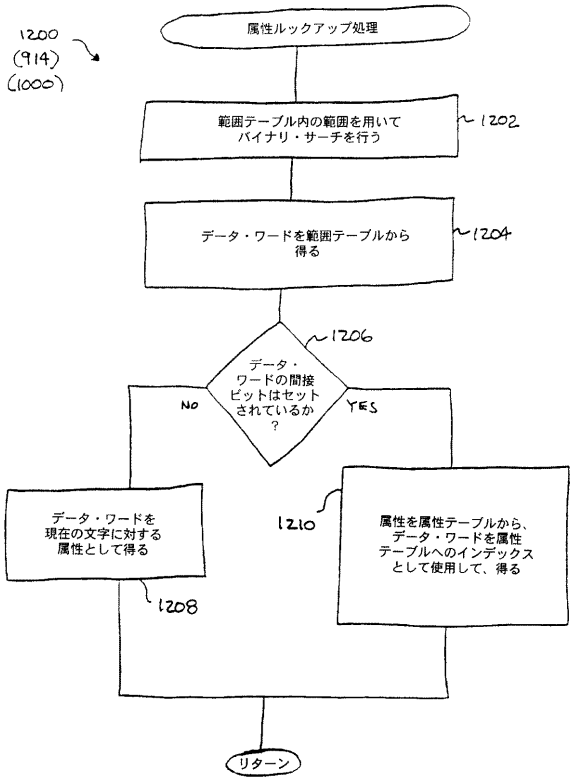
【図１１】



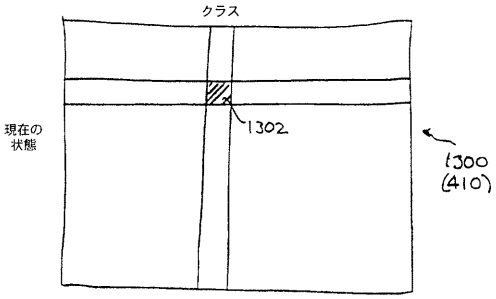
【図１２】



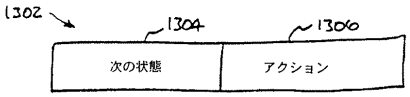
【図１３】



【図１４Ａ】



【図１４Ｂ】

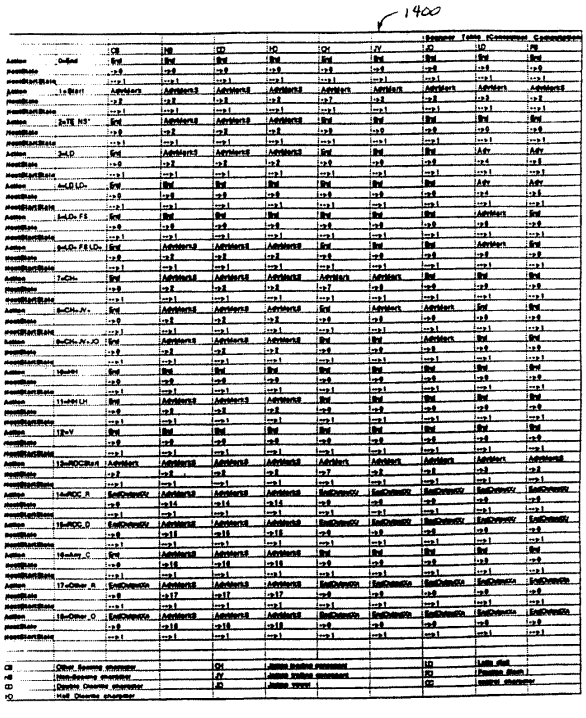


【図１５】

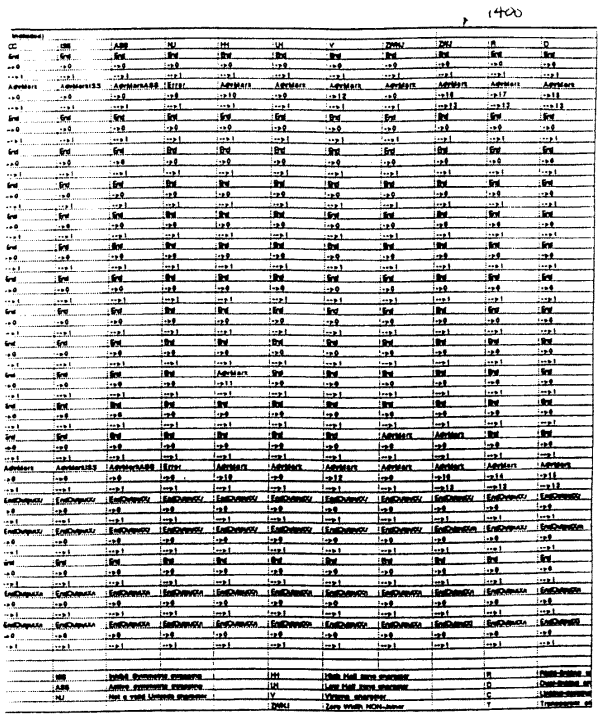
1400

状態	OS	NS	DD	HD	JLD	JVD	JF	LD	FS	CC	ISS	ASS	NU
0=End	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0
1=Start	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	AdvMark ->2	AdvMark ->2	AdvMark ->2	AdvMark ->2	AdvMark ->2	AdvMark ->0	AdvMark$ISS ->0	AdvMark$ASS ->0	Error ->0
2=TENS*	End ->0	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0
3=LD	End ->0	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	End ->0	End ->0	End ->0	Adv ->4	Adv ->5	End ->0	End ->0	End ->0	End ->0
4=LD LD+	End ->0	End ->2	End ->2	End ->2	End ->0	End ->0	End ->0	Adv ->4	Adv ->5	End ->0	End ->0	End ->0	End ->0
5=LD+ FS	End ->0	End ->2	End ->2	End ->2	End ->0	End ->0	End ->0	AdvMark ->4	AdvMark ->5	End ->0	End ->0	End ->0	End ->0
6=LD+ FS LD+	End ->0	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	End ->0	End ->0	End ->0	AdvMark ->4	End ->5	End ->0	End ->0	End ->0	End ->0
7=JL+	End ->0	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	AdvMark ->7	AdvMark ->8	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0
8=JL+JV+	End ->0	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	End ->0	AdvMark ->8	AdvMark ->9	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0
9=JL+JV+JF+	End ->0	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	AdvMark$ ->2	End ->0	AdvMark ->9	End ->0	End ->0	End ->0	End ->0	End ->0	End ->0

【図１６Ａ】

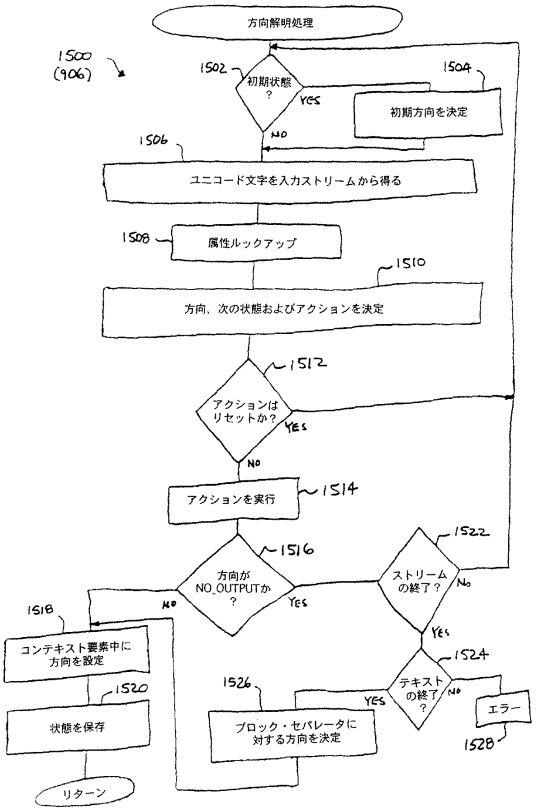


【図１６Ｂ】



16A 16B

【図１７Ａ】



【図１７Ｂ】

		sDIR	START	SL	SR	AL	L+AN	L+EN	
LR		START,L	SL,L	SL,L	SL,L	SL,L	SL,L	SL,L	
RL		START,R	SR,R	SR,R	SR,R	SR,R	SR,R	SR,R	
AL		START,R	AL,L	AL,R	AL,R	AL,R	AL,R	AL,R	
LRE		START,L	SL,push,L	SL,push,L	SL,push,L	SL,push,L	SL,push,L	SL,push,L	
RLE		START,R	SR,push,R	SR,push,R	SR,push,R	SR,push,R	SR,push,R	SR,push,R	
LRO		START,L	OR,push,L	OR,push,L	OR,push,L	OR,push,L	OR,push,L	OR,push,L	
RLO		START,R	OR,push,R	OR,push,R	OR,push,R	OR,push,R	OR,push,R	OR,push,R	
PDF		error	S(G),pop,G	S(G),pop,G	S(G),pop,G	S(G),pop,G	S(G),pop,G	S(G),pop,G	
AN	(G)+AN,L	sDIR,•	OR,G	L+AN,L	R+AN,L	AL+Num,L	L+AN,L	L+AN,L	
EN	(G)+EN,L	sDIR,•	OR,G	L+EN,L	R+EN,L	AL+Num,L	L+EN,L	L+EN,L	
ET	s#+ET,•	sDIR,•	OR,G	sL+N,•	sR+ET,•	sA+N,•	sLAN+T,•	L+EN,L	
ES	s(G)+N,•	sDIR,•	OR,G	sL+N,•	sR+N,•	sA+N,•	sLAN+N,•	sLEN+N,•	
CS	s(G)+N,•	sDIR,•	OR,G	sL+N,•	sR+N,•	sA+N,•	sLAN+S,•	sLEN+N,•	
ON	s(G)+N,•	sDIR,•	OR,G	sL+N,•	sR+N,•	sA+N,•	sLAN+N,•	sLEN+N,•	
BS	S(G),G	sDIR,•	reset	reset	reset	reset	reset	reset	
Notes:		scan for base direction	override	strong left	strong right	arabic letter	strong left+ arabic num	strong left+ euro num	

1511

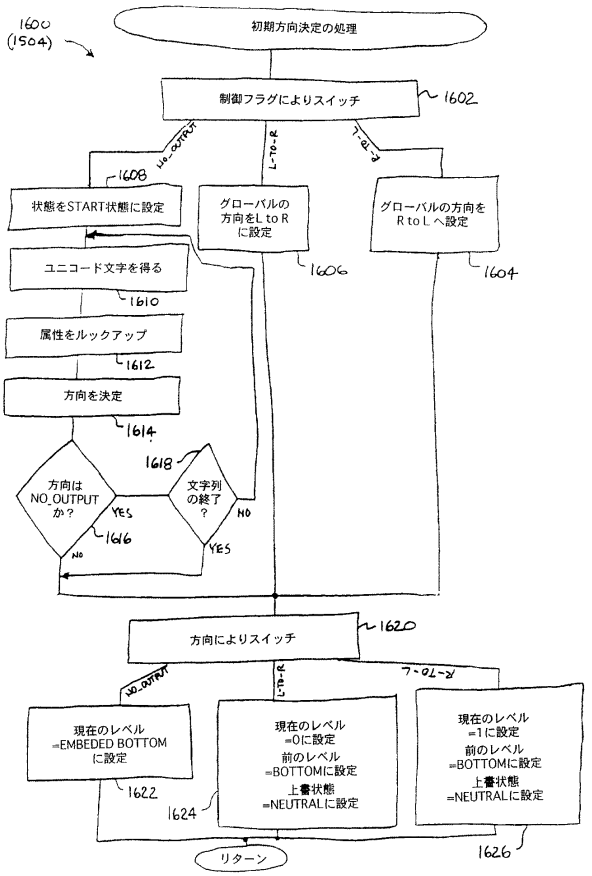
17B 17C 17D

【図１７Ｄ】

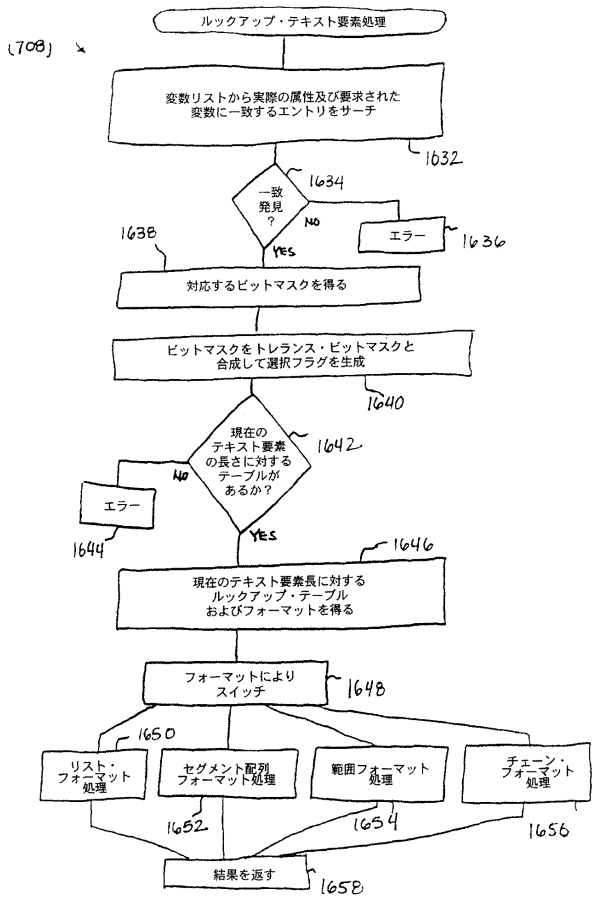
sLEN+N	sRAN+N	sREN+N	sALN+N	sLAN+T	sRAN+T	sLAN+S	sREN+S	sALN+S
L+EN,L	R+AN,G	R+EN,G	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+EN,G	AL+Num,G
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,L	R+AN,G	R+EN,G	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+EN,G	AL+Num,G
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,R	R+EN,R	AL+Num,R
L+EN,L	R+AN,G	R+EN,G	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+EN,G	AL+Num,G
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,G	R+EN,R	AL+Num,R
L+EN,G	R+AN,G	R+EN,G	AL+Num,G	L+AN,G	R+AN,R	L+AN,G	R+EN,G	AL+Num,G
L+EN,G	R+AN,R	R+EN,R	AL+Num,R	L+AN,R	R+AN,R	L+AN,L	R+EN,L	AL+Num,L
L+EN,L	R+AN,R	R+EN,R	AL+Num,R	L+AN,L	R+AN,G	L+AN,G	R+EN,L	AL+Num,L
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+T,•	sRAN+T,•	sLAN+N,•	sREN+N,•	sALN+N,•
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+N,•	sRAN+N,•	sLAN+N,•	sREN+S,•	sALN+S,•
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+N,•	sRAN+N,•	sLAN+S,•	sREN+S,•	sALN+S,•
sLEN+N,•	sRAN+N,•	sREN+N,•	sALN+N,•	sLAN+N,•	sRAN+N,•	sLAN+N,•	sREN+N,•	sALN+N,•
L+EN,G	R+AN,G	R+EN,G	AL+Num,G	L+AN,G	R+AN,G	L+AN,G	R+EN,G	AL+Num,G
				euro terms	lookaheads for numeric	separators		
						punctuation		

1511

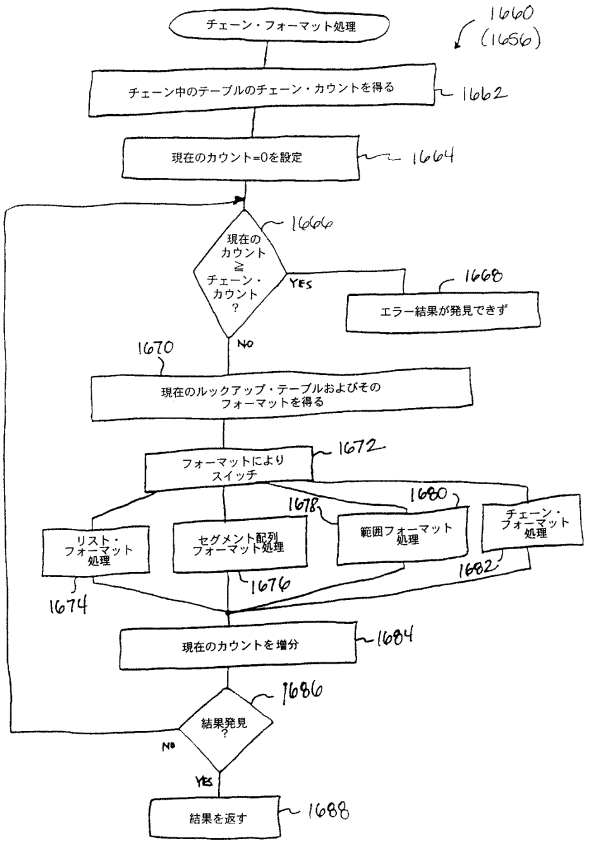
【図１８】



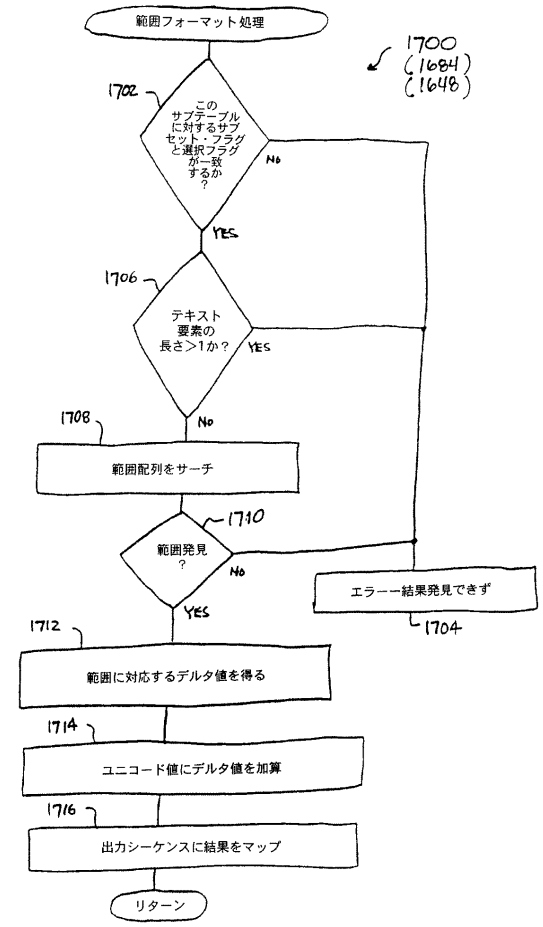
【図１９】



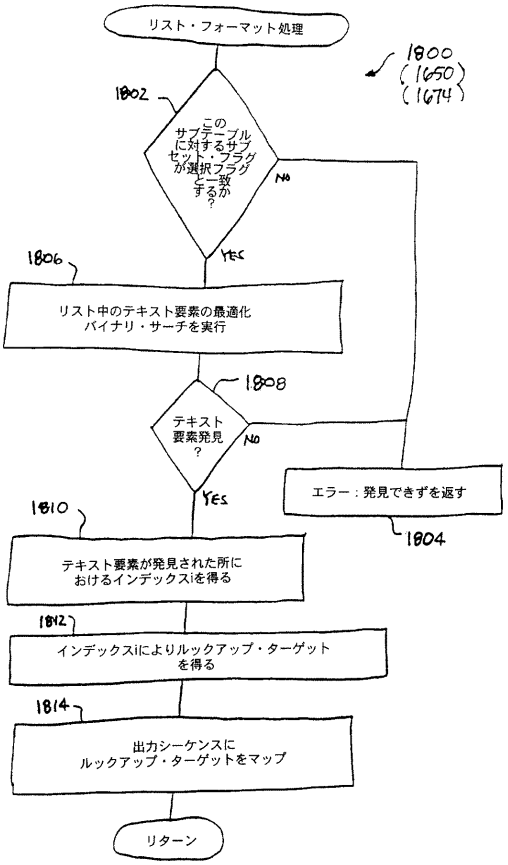
【図２０】



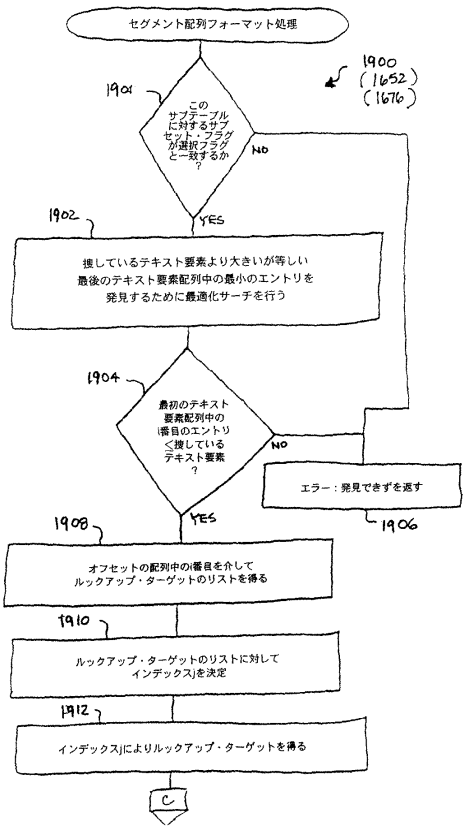
【図２１】



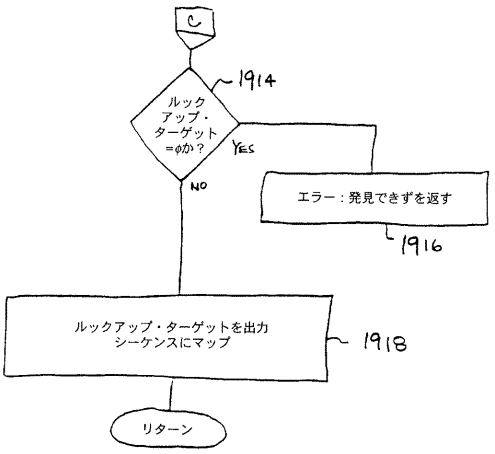
【図２２】



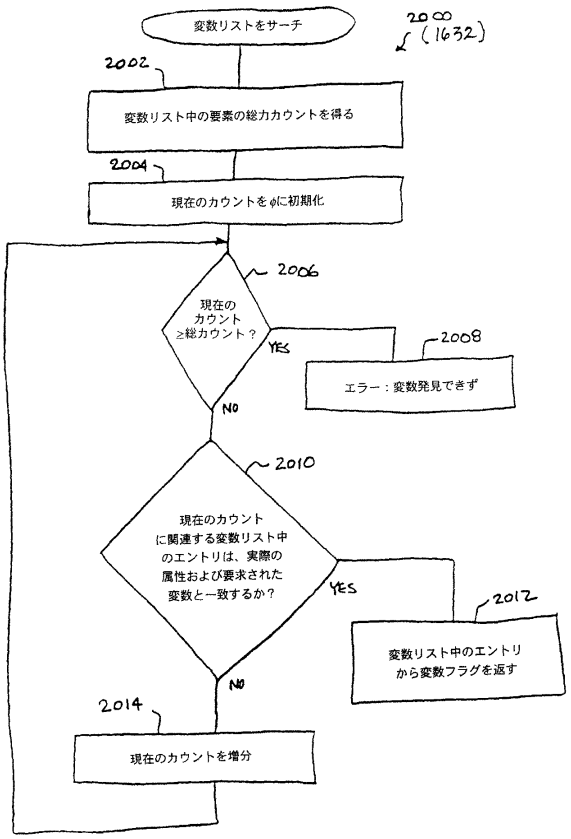
【図２３Ａ】



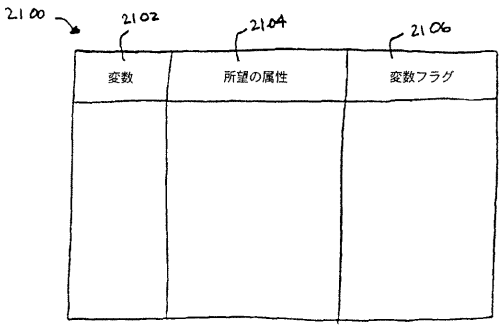
【図２３Ｂ】



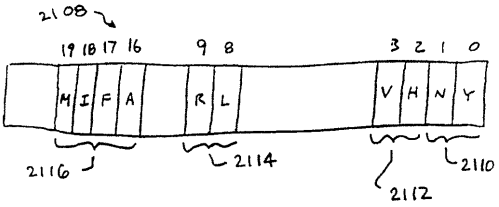
【図２４】



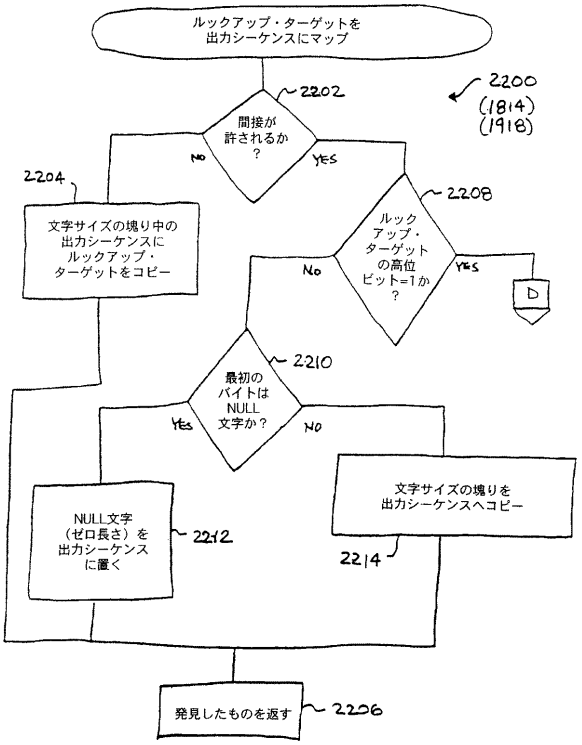
【図２５Ａ】



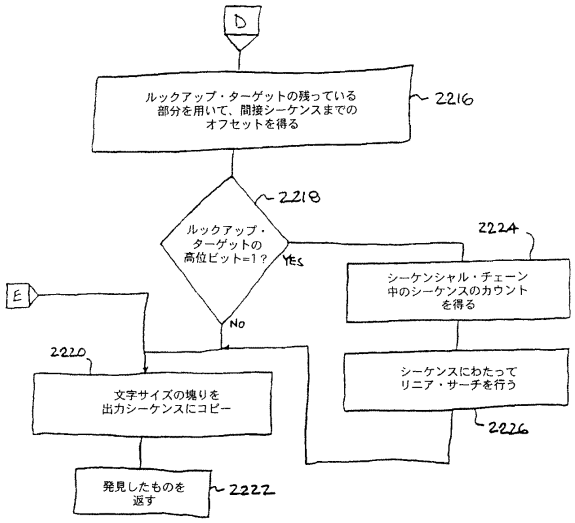
【図２５Ｂ】



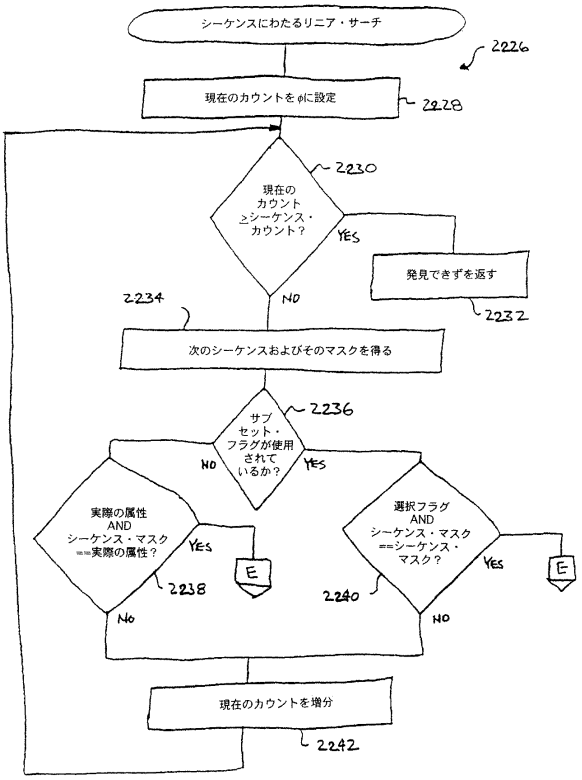
【図２６Ａ】



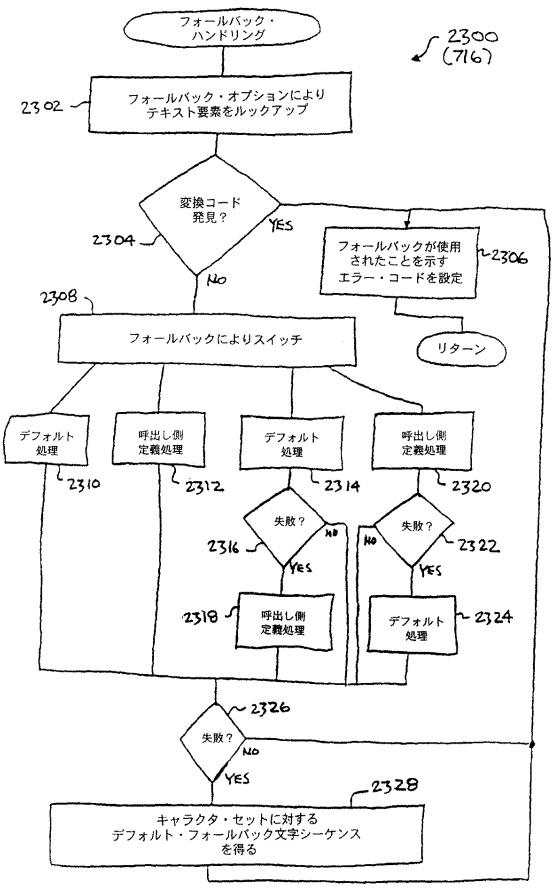
【図２６Ｂ】



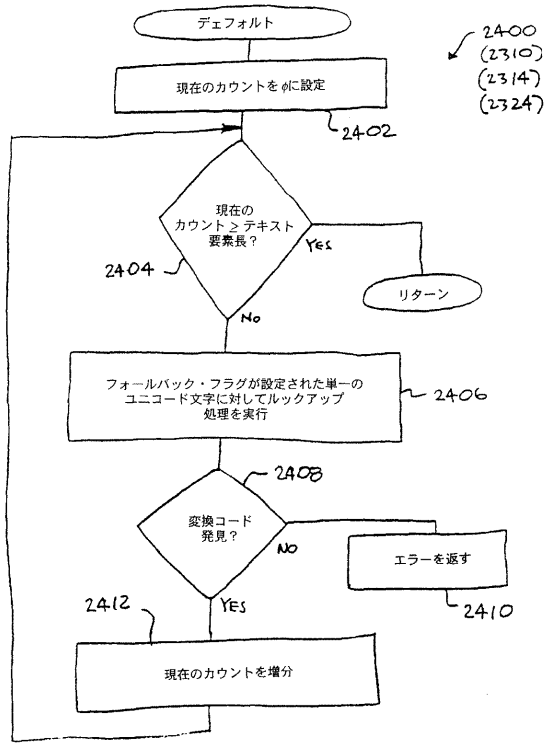
【図２６Ｃ】



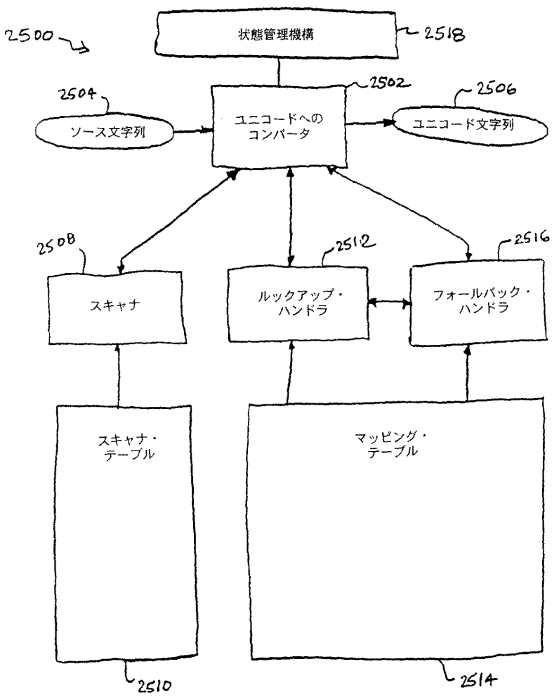
【図２７】



【図２８】



【図２９】



フロントページの続き

(31)優先権主張番号 08/527,837
(32)優先日 平成7年9月13日(1995.9.13)
(33)優先権主張国 米国(US)
(31)優先権主張番号 08/527,831
(32)優先日 平成7年9月13日(1995.9.13)
(33)優先権主張国 米国(US)

早期審理対象出願

(72)発明者 マコンネル，ジョン，アイ．
アメリカ合衆国 ９４０２５ カリフォルニア州 メンロ パーク マッケンドリー ドライブ ２２２
(72)発明者 タン，ユン－フォン，フランク
アメリカ合衆国 ９４０８２ カリフォルニア州 サニーヴェイル イースト ホームステッド ロード １７５ アパートメント ナンバー８
(72)発明者 ダニエルズ，アンドリュー，エム
アメリカ合衆国 ９４０２５ カリフォルニア州 メンロ パーク ハイト ストリート １２３

合議体
審判長 田口 英雄
審判官 飯田 清司
審判官 真木 健彦

(56)参考文献 特開平６－８３５７１（ＪＰ，Ａ）
特開平７－９３３７４（ＪＰ，Ａ）
特開平４－２３８５３１（ＪＰ，Ａ）
特開平７－１２１５１１（ＪＰ，Ａ）
特開平８－１６３６０（ＪＰ，Ａ）
特開平３－２２３９２３（ＪＰ，Ａ）