(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 November 2002 (28.11.2002)

PCT

(10) International Publication Number
WO 02/095950 A1

(51) International Patent Classification[7]: H03M 7/30

(21) International Application Number: PCT/US02/11989

(22) International Filing Date: 15 April 2002 (15.04.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/855,127 14 May 2001 (14.05.2001) US

(71) Applicant *(for all designated States except US)*: **UNISYS CORPORATION** [US/US]; Township Line and Union Meeting Roads, P.O. Box 500, Blue Hell, PA 19424-0001 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **COOPER, Albert, B.** [—/US]; 20 East 74th Street, Apt. 14C, New York, NY 10021 (US).

(74) Agents: **STARR, Mark, T.** et al.; Unisys Corporation, Township Line and Union Meeting Roads, P.O. Box 500, Blue Bell, PA 19424-0001 (US).

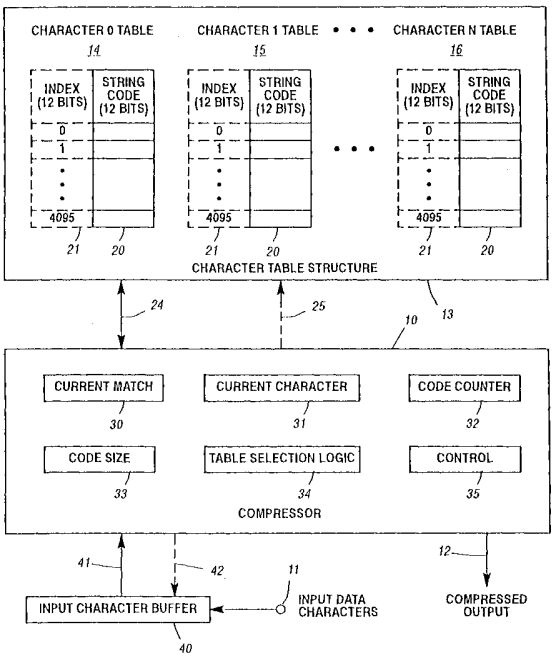
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: CHARACTER TABLE IMPLEMENTED DATA COMPRESSION METHOD AND APPARATUS



(57) Abstract: A new LZW compressor implementation architecture utilizes a plurality of character tables corresponding to the respective characters of the alphabet. A string is stored by storing the code associated with the string in the character table corresponding to the extension character of the string at a character table location corresponding to the code of the string prefix. A search for the longest matching string is performed by determining if the character table location is empty corresponding to the code of the currently matched string in the character table associated with the currently matched string in the character table associated with the currently fetched character. If the location is not empty, it is storing the code of the string comprising the currently matched string extended by the currently fetched character. This string code is used as the next match with which to continue the search with the next fetched character. When the location is empty, the longest match has been determined to be the currently matched string. The stored strings are updated by storing the next available string code in the empty location.

WO 02/095950 A1

CHARACTER TABLE IMPLEMENTED DATA COMPRESSION METHOD AND APPARATUS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to LZ data compression systems particularly with respect to the LZW compression methodology. More particularly, the invention relates to the architecture and protocols for storing and accessing data character strings in the compressor.

2. Description of the Prior Art

Professors Abraham Lempel and Jacob Ziv provided the theoretical basis for LZ data compression and decompression systems that are in present day widespread usage. Two of their seminal papers appear in the IEEE Transactions on Information Theory, IT-23-3, May 1977, pp. 337-343 and in the IEEE Transactions on Information Theory, IT-24-5, September 1978, pp. 530-536. A ubiquitously used data compression and decompression system known as LZW, adopted as the standard for V.42 bis modem compression and decompression, is described in U.S. Patent 4,558,302 by Welch, issued December 10, 1985. LZW has been adopted as the compression and decompression standard used in the GIF image communication protocol and is utilized in the TIFF image communication protocol. GIF is a development of CompuServe Incorporated and the name GIF is a Service Mark thereof. A reference to the GIF specification is found in GRAPHICS INTERCHANGE FORMAT, Version 89a, 31 July, 1990. TIFF is a development of Aldus Corporation and the name TIFF is a Trademark thereof. Reference to the TIFF specification is found in TIFF, Revision 6.0, Final - June 3, 1992.

Further examples of LZ dictionary based compression and decompression systems are described in the following U.S. patents: patent 4,464,650 by Eastman et al., issued August 7, 1984; patent 4,814,746 by Miller

et al., issued March 21, 1989; patent 4,876,541 by Storer, issued October 24, 1989; patent 5,153,591 by Clark, issued October 6, 1992; patent 5,373,290 by Lempel et al., issued December 13, 1994; patent 5,838,264 by Cooper, issued November 17, 1998; and patent 5,861,827 by Welch et al., issued January 19, 1999.

In the above dictionary based LZ compression and decompression systems, the compressor and decompressor dictionaries may be initialized with all of the single character strings of the character alphabet. In some implementations, the single character strings are considered as recognized although not explicitly stored. In such systems the value of the single character may be utilized as its code and the first available code utilized for multiple character strings would have a value greater than the single character values. In this way the decompressor can distinguish between a single character string and a multiple character string and recover the characters thereof. For example, in the ASCII environment, the alphabet has an 8 bit character size supporting an alphabet of 256 characters. Thus, the characters have values of 0-255. The first available multiple character string code can, for example, be 258 where the codes 256 and 257 are utilized as control codes as is well known.

In the prior art dictionary based LZ compression systems, data character strings are stored and accessed in the compressor dictionary utilizing well known searchtree architectures and protocols. Typically, the searchtree is arranged in nodes where each node represents a character, and a string of characters is represented by a node-to-node path through the tree. When the input character stream has been matched in the dictionary tree up to a matched node, a next input character is fetched to determine if the string match will continue. Conventionally, a determination is made to ascertain if the fetched character is already stored as an extension

node of the matched node. Various techniques are utilized to effect this determination such as hashing and sibling lists as are well understood in the art.

Although the known dictionary architecture and protocols provide efficient data compression systems, it is a continuing objective in the art to improve compressor performance.

SUMMARY OF THE INVENTION

The present invention provides a new string storage and access architecture and protocols which, it is believed, will improve the performance of LZ type data compression algorithms.

In the present invention a plurality of character tables corresponding to the respective characters of the alphabet are utilized instead of the conventional searchtree structured dictionary. A string is stored by storing the code associated with the string in the character table corresponding to the extension character of the string at a character table location corresponding to the code of the string prefix. The input data character stream is compared to the stored strings by determining if the table location corresponding to the code of the currently matched string in the character table associated with the currently fetched character is empty. If the location is not empty it is storing the code of the string comprising the currently matched string extended by the currently fetched character. This string code is utilized as the next match with which to continue the search with the next fetched character. If, however, the location is empty, the longest match has been determined to be the currently matched string, and the code thereof is output. The stored strings are updated by storing the next available string code in the empty location. The current character utilized to select the character table in which the empty location was encountered is the mismatching character that caused

the string matching process to terminate at the longest match. In an LZW embodiment, the mismatching character is utilized to begin the next string search by using this character as the initial current match for the new string.

An alternative embodiment of the invention includes creating a character table when the character corresponding thereto is first encountered in the input.

A still further embodiment involves storing the code of the string prefix together with the code of the string at a character table location and creating the table locations as update extended strings are encountered.

The present invention provides a new implementation architecture for data compression algorithms, such as LZW, that is believed will result in significant advantages such as enhanced speed and performance.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic block diagram of a data compressor for compressing data in accordance with the present invention. In the embodiment of Figure 1 the Character Tables are pre-established. The data compressor of Figure 1 is particularly suitable for implementation as a software embodiment.

Figure 2 is a control flow chart illustrating the operations executed by the compressor of Figure 1 so as to perform data compression in accordance with the present invention.

Figure 3 is a chart exemplifying the operations of the compressor of Figure 1 in accordance with the control flow chart of Figure 2.

Figure 4 is a schematic block diagram of an alternative embodiment of a data compressor for compressing data in accordance with the present invention. The data compressor of Figure 4 is particularly suitable

for implementation as a hardware embodiment.

Figure 5 is a schematic block diagram similar to that of Figure 4 providing an alternative configuration for determining the empty status of a Character Table location.

Figure 6 is a schematic block diagram of an alternative embodiment of a data compressor for compressing data in accordance with the present invention. In the embodiment of Figure 6, the Character Tables and the locations thereof are established when required. The data compressor of Figure 6 is particularly suitable for implementation as a software embodiment.

Figure 7 is a control flow chart illustrating the operations executed by the compressor of Figure 6 so as to perform data compression in accordance with the present invention.

Figure 8 is a schematic block diagram similar to that of Figure 1 with frequently selected Character Tables stored in cache memory.

Figure 9 is a schematic block diagram illustrating optional features for inclusion in the embodiments of the invention described herein.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The best mode embodiments described below, utilizing the Character Table implementation architecture of the present invention, are predicated, generally, on the LZW methodology. The embodiments utilize an implementation feature similar to that described above where the single character strings are considered as recognized by the compressor although not explicitly initialized therein.

Referring to Figure 1, a data compressor 10 is illustrated that compresses a stream of input data characters applied at an input 11 into a stream of corresponding compressed codes at an output 12. Included, in accordance with the invention, is Character Table

Structure 13 comprising Character Tables 14-16 corresponding to the respective data characters of the alphabet over which compression is being performed. Each of the Character Tables 14-16 is comprised of a plurality of Table locations 20 for storing the respective string codes of data character strings stored in the Character Table Structure 13. The Table locations 20 of the Character Tables 14-16 are accessed by respective Indices 21.

A data character string is comprised of a prefix string of one or more characters followed by an extension character. A string is stored in the Character Table Structure 13 by storing the string code associated with the string in the Character Table 14-16 corresponding to the extension character of the string at a Character Table location 20 corresponding to the code of the string prefix. The prefix code is utilized as the Index 21. Data is communicated between the compressor 10 and the Character Table Structure 13 via a bi-directional data bus 24 under control of a control bus 25.

In typical LZW data compressors, the dictionary is limited to 4096 string codes. When this limitation is applied to the present invention, the prefix code Indices 21 and String Code locations 20 will each be 12 bits wide with the Indices 21 ranging from 0 through 4095 as indicated. It is appreciated that when accounting for control codes, the Indices and Table locations corresponding to the control codes are not utilized. It is furthermore appreciated that with an N character alphabet, the first N Indices of each Character Table 14-16 correspond to the N respective characters. The first N Table locations are indexed by the respective character values. For example, in the ASCII environment the alphabet has an 8 bit character size supporting an alphabet of 256 characters having values of 0-255. The first available multiple character string code may, for example, be 258 where the codes 256 and 257 are utilized

as control codes.

The compressor 10 includes a Current Match register 30, a Current Character register 31, a Code Counter 32 and a Code Size register 33. The Code Counter 32 sequentially generates code values to be assigned to data character strings stored in the Character Table Structure 13 in a manner to be described. The Code Size register 33 is utilized, as is well known, to control the number of bits utilized for transmitting the compressed code from the output 12. Also included is table selection logic 34 for selecting the appropriate Character Table 14-16 in accordance with Current Character in a manner to be explained. The compressor 10 additionally includes control 35 for controlling the operations of the compressor 10 in accordance with the operational flow chart of Figure 2 to be described below.

Further included is an Input Character Buffer 40 that buffers the input data character stream received at the input 11. The input data characters are applied from the Input Character Buffer 40 via a bus 41 to the Current Character register 31 and the Current Match register 30 in accordance with operations to be described. The compressor 10 controls acquiring input data characters from the Input Character Buffer 40 via a control bus 42.

Briefly, the operation of the compressor 10 is as follows. The compressor 10 is initialized by setting the Code Counter 32 to the first available multiple character string code and by setting the Code Size register 33 to the initial Code Size. Additionally, the Current Match register 30 is cleared and the Character Tables 14-16 are cleared to empty. A first input data character is fetched to the Current Match register 30 to initiate a compression cycle. To begin a subsequent compression cycle, the Current Match register 30 is set to contain the mismatching character determined from the preceding compression cycle.

At the beginning of a compression cycle, the

next data character is fetched to the Current Character register 31. The Character Table 14-16 corresponding to Current Character is selected and the location corresponding to Current Match in the selected Character Table is accessed by Current Match utilizing the Indices 21. If the accessed location is not empty, the Current Match register 30 is set to the string code contained in the accessed location. The fetching of the next Current Character, the selection of the Character Table corresponding to Current Character and the accessing of the location corresponding to Current Match in the selected Character Table continues until the accessed location is empty.

When the accessed location is empty, the code in the Code Counter 32 is stored in the empty location and the code in the Current Match register 30 is output as the longest match. The Current Match register 30 is set to the mismatching character in the Current Character register 31 and the Code Counter 32 is incremented to the next available code. Control returns to fetch the next data character to the Current Character register 31 to begin the search for the next longest match in the next compression cycle.

Referring to Figure 2, with continued reference to Figure 1, a control flow chart is illustrated showing the detailed operations to be executed by the compressor 10. The control 35 of the compressor 10 is considered as containing appropriate circuitry, such as state machines, or appropriate software, to control execution of the operations. The flow chart of Figure 2 is predicated on a variable length output and the Code Size register 33 is utilized to this effect. In an ASCII variable length code implementation, the Code Size may begin with 9 bits and sequentially increase to 10, 11 and 12 bits at codes 512, 1024 and 2048, respectively. It is appreciated that a fixed code size may also be utilized with appropriate modifications to the embodiment.

Control enters a block 50 whereat the Code Counter 32 is initialized to a first available code, for example, 258 in the ASCII environment. At a block 51, the Code Size register 33 is initialized to the beginning Code Size, for example, 9 bits in ASCII embodiments. At a block 52, the Current Match register 30 is cleared and the Character Tables 14-16 are cleared to empty. Zero may be utilized in the locations 20 of the Character Tables to denote the empty state. At a block 53, the first input data character is fetched to the Current Match register 30 and, at a block 54, the next input data character is fetched to the Current Character register 31.

At a block 55, the Character Table 14-16 corresponding to Current Character is selected and, at a block 56, the location corresponding to Current Match in the selected Character Table is accessed via the Indices 21. The Table selection logic 34 is utilized to perform the Table selection. At a block 57, if the accessed location in the selected Character Table is not empty, the NO branch from the block 57 is taken to a block 58. At the block 58, the Current Match register 30 is set to the string code in the accessed location of the selected character table. Thereafter, control returns to the block 54 to fetch the next input data character to the Current Character register 31.

If, at the block 57, the accessed location in the selected Character Table is empty, the YES branch from the block 57 is taken to a block 70. When this occurs, the longest matching string in the Character Table Structure 13 has been determined. At the block 70, the code in the Code Counter 32 is stored in the empty accessed location of the selected Character Table thereby storing the appropriate extended string.

Control proceeds to a block 71 whereat the code of the Current Match is output as part of the compressed code stream provided at the compressor output 12. The

code of the Current Match is provided by the Current Match register 30 and is output utilizing the number of bits denoted by the Code Size register 33. When Current Match is a multiple character string, the code of the string resides in the Current Match register 30 and was the longest match found in the Character Table Structure 13 as described above with respect to the block 57. It is appreciated that the Current Match output at the block 71 can also be a single character. The output code in this case is the value of the character which is also provided from the Current Match register 30.

Processing proceeds to a block 72 whereat the character in the Current Character register 31 is set into the Current Match register 30. Thus, the Current Match register 30 is set with the character that resulted in the mismatch at the block 57.

Processing then proceeds to a block 73 whereat the code in the Code Counter 32 is tested to determine if an increase in code size is required. If so, processing continues to a block 74 whereat the Code Size register 33 is incremented by 1. If an increase in Code Size is not required at the block 73, the block 74 is bypassed to continue processing at a block 75. At block 75, the Code Counter 32 is incremented by 1. Control then returns to the block 54 to begin the next compression cycle with the mismatching character set into the Current Match register 30.

It is appreciated from the foregoing that the loop comprising the blocks 54-58 sequentially fetch the input data characters which select the corresponding Character Tables to determine if the strings represented by Current Match are stored in the Character Table Structure. When the YES branch is taken from the block 57, the longest matching string has been determined with the code thereof in the Current Match register and the mismatching character in the Current Character register. The extended string comprising the longest match extended

by the mismatching character is readily stored in the Character Table Structure by, at the block 70, inserting the code in the Code Counter into the empty location accessed by Current Match in the Character Table selected by Current Character.

Referring to Figure 3, with continued reference to Figures 1 and 2, an example of the operation of the compressor 10 in accordance with the flow chart of Figure 2 is illustrated. At the top of Figure 3, an input data character stream is shown where sequential characters are identified by character sequence numbers. This is done to facilitate following the progress of the characters through the steps of the example. It is appreciated that the sequence numbers are shown for purposes of character identification and do not appear in the actual data character stream.

The example is largely self-explanatory, with the actions performed delineated in the left-hand column and the blocks of Figure 2 that participate in the actions designated in the right-hand column. The selected Character Table corresponding to Current Character is indicated in the Character Table column with the Current Character value denoted in the Current Character column. The Character Table Indices and the Character Table String Code fields are illustrated in the respective columns as indicated. When, at the block 57 of Figure 2, an accessed Character Table location is not empty, the string code stored therein is indicated by parenthesis. The Index column of the Character Table indicates, in parenthesis, that the prefix code of the stored string is utilized to index or access the location of the selected Character Table. It is appreciated that the Index is provided by Current Match.

The operational example of Figure 3 graphically demonstrates the novel data compression implementation architecture of the present invention for storing data character strings, searching the input stream for the

longest match with the stored strings and updating the stored strings with an extended string. For example, action 1 illustrates the string "ab" stored in the "b" Character Table with a string code of 258. As described above with respect to Figures 1 and 2, the fetched Current Character "b" selects the "b" Character Table which is indexed by the string prefix code "a" (Current Match) and the string code 258 from the Code Counter is stored at the accessed location of the selected Character Table. The string is stored as described at the block 70 of Figure 2 because at the block 57 the accessed location of the selected Character Tables was determined to be empty.

In action 3, this string "ab" is encountered in the "b" Character Table and, at action 4, is extended in the "a" Character Table which stores the string "aba". As illustrated in action 4, this string is assigned the string code 260 from the Code Counter. In action 3, the string was encountered as being previously stored because, at the block 57, the accessed location in the selected Character Table was not empty. As seen in action 4, the string code 258, indicated in parenthesis in action 3, is set into the Current Match register at the block 58 of Figure 2.

In the embodiments described herein, when the last input data character has been fetched, the value in the Current Match register 30 is output as the last compressed code. Thus, in action 19, the last character of the exemplified input data character stream is output as illustrated.

More detailed descriptions of the actions of Figure 3 relative to the blocks of Figure 2 are readily apparent and will not be provided for brevity.

Referring to Figure 4, with continued reference to Figures 1-3, an alternative best mode embodiment of the data compressor of the present invention is illustrated. The embodiment of Figure 4 is particularly

suitable for hardware implementation. Many of the components of the Figure 4 embodiment correspond to components of the Figure 1 embodiment and are provided with reference numerals that are greater by 100 than the reference numerals of the corresponding components of Figure 1. Additionally, further components of Figure 4 correspond to blocks of Figure 2 and are provided with reference numerals that are also greater by 100 than the reference numerals of the corresponding blocks of Figure 2. The descriptions given above with respect to Figures 1 and 2 also apply to the structure and operation of these components in Figure 4. In addition, the compressor of Figure 4 includes select circuits 180 and 181 as well as Table Output Buffers 184-186.

Although the structural and functional descriptions given above with respect to Figures 1 and 2 also apply to the structure and operation of the compressor of Figure 4, a significant additional feature is included in the compressor of Figure 4 for selecting a Character Table 114-116 in accordance with Current Character in the Current Character register 131. In the compressor of Figure 4, the Current Match in Current Match register 130 simultaneously accesses the Character Tables 114-116 via the Indices 121. Current Character in the Current Character register 131 selects the appropriate accessed string code output via the select circuit 180. The string code outputs from the Character Tables 114-116 are buffered in the Table Output Buffers 184-186, respectively.

Similarly, the Current Character in Current Character register 131 selects, via the select circuit 181, the Character Table 114-116 into which to store the code in the Code Counter 132 in the empty location accessed by Current Match in the Current Match register 130.

The step by step operation of the compressor of Figure 4 is as follows. The controller 135, which

is responsive to the Code Counter 132 and to the detector 157, which tests the selected Character Table location for the empty status, provides control inputs to all of the blocks of Figure 4 to control the operations thereof. The controller 135 may include state machines to sequence through the operations. Initially, the Code Counter 132 is initialized to the first available code via control input 150, the Current Match register 130 is cleared, and the Character Tables 114-116 are cleared to the empty state.

In the first compression cycle, the first input data character is fetched from the Input Character Buffer 140 to the Current Match register 130 via bus 153 and the next input data character is fetched, via bus 154, to the Current Character register 131. In a subsequent compression cycle, the Current Match register 130 is set to contain the mismatching character determined from the preceding compression cycle by transferring Current Character from the Current Character register 131 to the Current Match register 130 via the bus 172.

At the beginning of a compression cycle, the next input character is fetched to the Current Character register 131 and Current Match accesses, in parallel, the locations corresponding thereto in the Character Tables 114-116 via the Table Indices 121. The contents of the accessed String Code locations 120 are simultaneously read into the Table Output Buffers 184-186, respectively. The Character Table output corresponding to Current Character is selected by the select circuit 180 and is tested for empty by the detector 157. If the accessed location 120 in the selected Character Table is not empty, the controller 135 enables the select circuit 180 to set the Current Match register 130 to the Character Table String Code output selected by Current Character. The next input data character is then fetched from the Input Character Buffer 140 to the Current Character register 131 via the bus 154. The controller

135 controls the repetition of these operations until the detector 157 indicates that the Character Table location accessed by Current Match and selected by Current Character is empty.

When the accessed location is empty, the code in the Code Counter 132 is directed, by the select circuit 181, for storage in the String Code field of the Character Table selected by Current Character at the accessed location corresponding to Current Match. Thereafter, Current Match is output via the bus 171 utilizing the number of bits determined by the Code Size control 133. The controller 135 controls the Code Size control 133 to select the appropriate Code Size in accordance with the count in the Code Counter 132 in a well known manner.

To prepare for the next compression cycle, the Current Match register 130 is set to the mismatching character in the Current Character register 131, via the bus 172, and the Code Counter 132 is incremented to the next available code via the input 175.

It is appreciated that the operative example of Figure 3 is also applicable to the Figure 4 embodiment.

Referring to Figure 5, where like reference numerals indicate like components with respect to Figure 4 and with continued reference to Figures 1-4, an alternative best mode embodiment of the compressor of the present invention is illustrated. The embodiment of Figure 5 is substantially the same as the embodiment of Figure 4 and therefore operates as described above with respect to Figure 4 generally following the flow of operations depicted in Figure 2. The principle difference between the embodiments of Figures 4 and 5 is the manner in which the empty status of a Character Table location is detected. In the Figure 4 embodiment, the Character Tables were initially cleared to empty by, for example, setting all of the locations 120 to an empty indication such as zero. In the Figure 5 embodiment, a 1 bit Empty Flag 200 is included with each

String Code location 120 in each of the Character Tables 114-116. In the Figure 5 embodiment, it is only necessary to reset the Empty Flag 200 at each Character Table location in order to initialize the Character Tables to empty.

The Code Counter 132 includes a 1 bit extension 201 which is set to 1 so as to indicate the occupied status of a Character Table location. When an extended string is stored in the Character Table Structure 113, as described above, not only is the code in the Code Counter 132 written into the accessed location of the selected Character Table, as previously described, but the bit 201 is also written into the associated Empty Flag 200 setting the Flag to "occupied". The Character Table is selected by the select circuit 181 and the appropriate location therein accessed by the Current Match register 130 as described above.

In the Figure 5 embodiment, the test for empty at the block 57 of Figure 2, is now performed by a select circuit 202 coupled to the Current Character register 131. The select circuit 202 selects the Empty Flag 200 associated with the Character Table location accessed by Current Match from the Character Table selected by Current Character. The select circuit 202 provides an input to controller 203 so as to effect the empty test and control the resulting operations as described above. It is appreciated that the controller 203 performs the functions of the controller 135 described above with respect to Figure 4. It is further appreciated that the operational example of Figure 3 also applies to the Figure 5 embodiment.

As discussed above with respect to Figures 4 and 5, the Current Match register 130 is set to Current Character toward the end of a compression cycle thereby simultaneously accessing the Indices 121 of all of the Character Tables 114-116. The contents of the accessed Character Table locations are simultaneously read from

the Tables into the Table Output Buffers 184-186. Thus, at the beginning of a compression cycle, the appropriate values from the accessed String Code locations 120 reside in the Table Output Buffers 184-186 ready for selection via the select circuit 180 when Current Character is fetched into the Current Character register 131 at the beginning of the cycle. The compressors of Figures 4 and 5 are not required to wait for Character Table selection and Character Table read-out after Current Character is fetched. It is believed that this arrangement will effect an even further enhancement of compressor speed.

Referring to Figure 6, in which like reference numerals indicate like components with respect to Figure 1 and with continued reference to Figures 1-3, an alternative best mode embodiment of the data compressor of the present invention is illustrated. Although the embodiment of Figure 6 is configured and operates in a manner similar to that described above with respect to Figures 1 and 2, the Figure 6 embodiment includes significant additional features that provide further enhancements.

The embodiment of Figures 1 and 2, as well as the embodiments of Figures 4 and 5, utilize pre-established Character Tables as described. In the Figure 6 embodiment, a Character Table is established when the character corresponding thereto is first encountered in the input. As a further feature, a location in a Character Table is established when the location is required in which to store the above described extended string. By utilizing these additional features, only Character Table memory actually utilized by the operation of the compressor is required.

Accordingly, the Figure 6 embodiment includes Character Table Structure 300 illustrating Character Tables 301-303. Each Character Table 301-303 is established when the input data character corresponding

thereto is first encountered in the Current Character register 31. In order to facilitate implementation of the additional features of the Figure 6 embodiment, a Character Table 301-303 includes a location 306 in which to store the code of a string and a corresponding location in which to store the code of the prefix of the string. The stored Prefix Codes provide a Prefix Code List 307 with which to index and access the corresponding String Codes 306.

The embodiment of Figure 6 includes compressor 310 comprising the previously described Current Match register 30, the Current Character register 31, the Code Counter 32 and the Code Size register 33. In addition, the compressor 310 includes table creation and selection logic 311 and Character Table Links 312 utilized in the "on-the-fly" creation of the Character Table Structure 300. The Character Table Links 312 may be configured as a directory indexed by the individual characters of the alphabet to provide a respective address link to the corresponding Character Table in the Character Table Structure 300. If a link is not stored for an input data character, the Character Table for that character has not yet been established. When an input data character is first encountered by the compressor 310 and fetched into the Current Character register 31, the logic 311 selects an address link for storage in the directory 312 to correspond to the newly encountered character. The logic 311 then creates the Character Table linked by this address so as to provide the Character Table corresponding to the character.

The compressor 310 additionally includes control 313 for controlling the operations of the compressor 310 in accordance with the operational flow chart of Figure 7.

Referring to Figure 7, with continued reference to Figures 1, 2 and 6, a control flow chart is illustrated showing the detailed operations to be executed by the

compressor 310. The control 313 of the compressor 310 is considered as containing appropriate circuitry, such as state machines, or appropriate software, to control execution of the operations. The flow chart of Figure 7 is predicated on a variable length output in the manner described above with respect to Figure 2. The descriptions given above with respect to Figure 2 apply, where applicable, to the flow chart of Figure 7.

Control enters blocks 350-354 to perform functions similar to those described above with respect to blocks 50-54, respectively, of Figure 2. It is noted at block 352 that the Current Match register is cleared. The Character Tables that were cleared at block 52 of Figure 2 are not as yet established in the Figure 6 embodiment.

At a block 355, the compressor 310 determines if a Character Table has been established for Current Character. The Character Table Links 312 are consulted to effect this determination. If at the block 355, a Character Table has been established for Current Character, the YES branch is taken from the block 355 to a block 356. At the block 356, the Character Table corresponding to Current Character is selected utilizing the table creation and selection logic 311.

At a block 357, the Current Match is compared to the Prefix Code List 307 of the selected Character Table to determine if Current Match is in the Prefix Code List. The Prefix Code List may be searched associatively for Current Match or, alternatively, the Prefix Code List may comprise an ordered and linked list of Prefix Codes so as to effect the comparison with Current Match. Other list searching procedures known in the art may be utilized to the same effect. If Current Match is found in the Prefix Code List, the YES branch is taken from the block 357 to a block 358.

At the block 358, the location corresponding to Current Match in the selected Character Table is accessed and, at a block 359, the Current Match register

30 is set to the string code in the accessed location of the selected Character Table. Control then returns to the block 354 to fetch the next character to the Current Character register 31 to continue the search for the longest match.

It is appreciated that the loop comprised of the blocks 354-359 searches for the longest matching string in the Character Table Structure 300 generally as described above with respect to the blocks 54-58 of Figure 2. In the Figure 6 embodiment, when Current Match is found in the Prefix Code List at the block 357, the longest match is not yet determined. When, however, at the block 357, Current Match is not in the Prefix Code List of the selected Character Table, the longest match has been determined and the NO branch from the block 357 is taken to a block 370.

Accordingly, at the block 370, Current Match in the Current Match register 30 is added to the Prefix Code List 307 of the selected Character Table. Thus, at the block 370, a location is established in the selected Character Table that corresponds to Current Match. If the Prefix Code List comprises an ordered and linked list, the Current Match is inserted and linked into the Prefix Code List in the appropriate order. At a block 371, the code from the Code Counter 32 is stored in the String Code field at the location in the selected Character Table that was established at the block 370. It is appreciated that the block 371 of Figure 6 corresponds to the block 70 of Figure 2 in the operational flow whereat the appropriate extended string is stored.

Control proceeds from the block 371 to blocks 391-395 whereat the functions described above with respect to blocks 71-75 of Figure 2 are performed. After executing the function of block 395, control returns to the block 354 to fetch the next input character to the Current Character register 31 to begin the next

compression cycle.

If, at the block 355, a Character Table has not been established for Current Character, the NO branch from the block 355 is taken to a block 380. At the block 380 a Character Table corresponding to Current Character is established and Current Match is entered as the first entry in the Prefix Code List 307 of the established Character Table. This first entry thereby establishes a String Code location 306 corresponding to Current Match in the established Character Table. The table creation and selection logic 311 and the Character Table Links 312 are utilized, as described above, in establishing the Character Table.

At a block 381, the code from the Code Counter 32 is stored in the established String Code location 306 in the established Character Table. Thus the appropriate extended string is stored, as in the block 371, in the manner described above with respect to the block 70 of Figure 2. Control then proceeds to the blocks 391-395 as previously described.

It is appreciated that the operative example of Figure 3 is also applicable to the Figure 6 embodiment.

Although the Figure 6 embodiment was described in terms of "on-the-fly" creation of the Character Tables as well as of the locations thereof, it is appreciated that these features can also be used separately. If the Character Table establishing feature is separately utilized, a complete Character Table, as illustrated in Figure 1, may be established and indexed in the manner described above with respect to Figure 1. In such an embodiment, Character Table locations would be tested for the empty status as described with respect to block 57 of Figure 2 rather than utilizing the Prefix Code List as described with respect to Figures 6 and 7.

If the Character Table location establishing feature is separately utilized, the Character Table links of Figure 6 would not be used as described. Instead,

pre-established Character Tables with Prefix Code Lists would be utilized.

Referring to Figure 8, in which like reference numerals indicate like components with respect to Figure 1 and with continued reference to Figures 1-3, an alternative best mode embodiment of the data compressor of the present invention is illustrated. The Figure 8 embodiment is substantially the same as the Figure 1 embodiment except that the Character Table Structure is segregated into cached Character Tables and non-cached Character Tables.

Accordingly, a Character Table Structure 400 is comprised of a cached section 401 and a non-cached section 402. Data is communicated between the compressor 10 and the cached Character Tables 401 via a bi-directional data bus 403 under control of a control bus 404. Data is communicated between the compressor 10 and the non-cached Character Tables 402 via a bi-directional data bus 405 under control of a control bus 406.

The compressor 10 includes control 410 for controlling the execution of the operations to be performed by the Figure 8 embodiment in a manner similar to that described above with respect to Figures 1 and 2. In addition, control 410 controls the operations performed with respect to the cached and non-cached sections 401 and 402.

The cached section 401 is implemented by a relatively small fast cache memory of a type well known in the art. The non-cached section 402 may be implemented with a larger and slower memory arrangement. When the Figure 8 embodiment is utilized to compress data from a body of data of a genre where the frequency distribution of the alphabet characters are known or determinable, Character Tables corresponding to frequently encountered characters are established in the cached section 401 while Character Tables corresponding to the infrequently

encountered characters are established in the non-cached section 402. It is believed that an overall speed improvement will be effected by this arrangement.

The embodiment of Figure 8 can be made automatically and dynamically configurable by providing automatic statistical analysis of the input data and providing logic in the compressor 10 for automatically and dynamically maintaining Character Tables in the cached section 401 corresponding to those characters of the alphabet that are dynamically found to be frequently occurring.

It is appreciated that the operative example of Figure 3 is also applicable to the Figure 8 embodiment.

Referring to Figure 9, with continued reference to Figures 6 and 7, additions to the compressor 310 of Figure 6 are schematically depicted for controlling further desirable features. Specifically, the compressor 310 of Figure 9 includes string length limitation logic 320, Character Table exclusion logic 321 and Character Table length limitation logic 322.

The string length limitation logic 320 limits the length of the string that is stored in the Character Table Structure 300 of Figure 6. Appropriate character counting logic is included in the logic 320 for maintaining a count of the characters fetched at the block 354 of Figure 7. When the length of the input string being searched exceeds the string length limit, the blocks 391-395 of Figure 7 are executed without storing the extended string. The string length limitation logic 320 is utilized to exclude from storage excessively long strings that are not likely to reoccur.

The Character Table exclusion logic 321 is utilized to exclude creation of Character Tables corresponding to alphabet characters that are likely to occur infrequently. In this manner a string ending in an extension character that is unlikely to occur will not be stored in the Character Table Structure 300.

The logic 321 may be implemented by including functionality in Figure 7 associated with the block 355 that determines if Current Character is such an infrequently occurring an character. If so, a Character Table corresponding to Current Character is not created and the blocks 391-395 of Figure 7 are executed to complete the compression cycle. By the use of the logic 321, strings that end in an infrequently occurring character and are thus unlikely to be repeated, are excluded from storage.

The Character Table length limitation logic 322 is utilized to limit the length of the Character Tables 301-303 to respective predetermined numbers of locations. Counters would be included in the logic 322 to be used with the blocks 357 and 370 of Figure 7. Before a location is established in a Character Table at the block 370 of Figure 7, the appropriate Character Table location counter is consulted to determine if further space exists in the Character Table before the location is established. If the Character Table length limitation is exceeded, the location is not established and the string is not stored at the block 371. The blocks 391-395 of Figure 7 are then executed to conclude the compression cycle. The Character Table length limitations may be established on the basis of character frequency of occurrence criteria. Character Tables for frequently occurring characters can be longer than Character Tables for infrequently occurring characters.

Although the features illustrated in Figure 9 are described with respect to inclusion in the Figure 6 embodiment, it is appreciated that these features can be included in any of the above described embodiments of the invention.

It is appreciated that the compressed code output provided by each of the embodiments of Figures 1-8 is compatible with standard LZW decompressors and the data character stream corresponding to the compressed code

output can be recovered thereby. If a feature discussed above with respect to Figure 9 is included in the compressor, corresponding logic should be utilized at the decompressor.

It is further appreciated that when no further input data characters are available, the above described embodiments will output Current Match to conclude the processing.

The architecture of the present invention is particularly advantageous for use in implementing the LZW compression requirements of GIF. In the GIF protocols, the LZW character size is adjusted to match the size of the data character alphabet over which compression is being performed. Since, for example in the Figure 6 embodiment, Character Tables are created when the alphabet characters are first encountered, the Figure 6 embodiment provides automatic adaptation of the LZW process to the alphabet size. It is also advantageous to use the smaller number of Character Tables that would be commensurate with the smaller alphabets often encountered in GIF processing.

It is appreciated from the foregoing that the implementation architecture of the present invention should result in an improvement in compressor speed. The present invention utilizes unambiguous accessing of Character Table locations thereby eliminating the time required for collision resolution in prior art hashing implementations or prior art sibling list searches. Furthermore, the architecture of the present invention is particularly suited to parallel Character Table accessing as described above with respect to Figures 4 and 5. It is noted with respect to the embodiments of Figures 4 and 5, that the present invention is readily adaptable to hardware implementation thereby benefiting from the speed advantages inherent therein. It is also appreciated that further performance enhancements should be realized because of the implementation simplicity

evidenced in the above described embodiments.

Although the above described embodiments of the invention are LZW based, it is appreciated that the architecture of the present invention can be utilized with other known dictionary based compression methodologies. It is furthermore appreciated that since the architecture of the present invention is predicated on Character Tables corresponding to the respective characters of the alphabet, the input data characters can be over any size alphabet having any corresponding character bit size. For example, the data characters can be textual data, image pixel data or bit map data. The input data can also be binary characters over the two-character binary alphabet 1 and 0 having a 1-bit size character.

It is appreciated that the above described embodiments of the invention may be implemented in hardware, firmware, software or a combination thereof. Discrete circuit embodiments may readily be implemented for performing the various described functions. In a software embodiment, appropriate modules programmed with coding readily generated from the above descriptions may be utilized.

While the invention has been described in its preferred embodiments, it is to be understood that the words which have been used are words of description rather than of limitation and that changes may be made within the purview of the appended claims without departing from the true scope and spirit of the invention in its broader aspects.

CLAIMS

1. A data compression method for compressing an input stream of data characters into an output stream of compressed codes, said data characters being from an alphabet of data characters, comprising

providing a plurality of character tables corresponding to respective characters of said alphabet,

storing, in said character tables, strings of data characters encountered in said input stream, said stored strings having respective codes associated therewith, a string comprising a prefix string of at least one of said characters followed by an extension character,

a particular string being stored in said character tables by storing the code associated with said particular string in the character table corresponding to the extension character of said particular string at a character table location corresponding to the code of the prefix string of said particular string,

searching said input stream by comparing said input stream to said stored strings to determine the longest match therewith,

outputting the code associated with said longest match so as to provide said output stream of compressed codes, and

inserting an extended string into said character tables, said extended string comprising said longest match extended by the next data character in said input stream following said longest match, said extended string being stored in the character table corresponding to said next data character.

2. The compression method of claim 1 wherein said searching step includes

(a) matching one of said stored strings thereby providing a current match,

(b) fetching the next data character from said input stream following said current match thereby providing a current character,

(c) determining if the location corresponding to said current match in a character table corresponding to said current character is empty, and

(d) if said location is not empty, setting said current match to the code stored in said location and repeating steps (b) through (d) until the location of step (c) is determined to be empty, thereby determining said longest match.

3. The compression method of claim 2 wherein, if said location in step (c) is determined to be empty, said outputting step comprises outputting said current match.

4. The compression method of claim 2 wherein, if said location in step (c) is determined to be empty, said inserting step includes

providing a next available string code, and

storing said next available string code in said empty location,

thereby inserting said extended string into said character table corresponding to said current character.

5. The compression method of claim 3 wherein said compression method operates in compression cycles, further including

setting said current match to said current character in preparation for performing a next compression cycle.

6. The compression method of claim 1 wherein said searching step includes

(a) matching one of said stored strings thereby providing a current match,

(b) fetching the next data character from said input stream following said current match thereby providing a current character,

(c) accessing the locations corresponding to said current match in said plurality of character tables so as to provide a plurality of character table outputs corresponding to said accessed locations, respectively,

(d) selecting the character table output from said character table corresponding to said current character,

(e) determining if said selected character table output is empty, and

(f) if said selected character table output is not empty, setting said current match to the code stored in said selected character table output and repeating steps (b) through (f) until the selected character table output of step (e) is determined to be empty, thereby determining said longest match.

7. The compression method of claim 6 wherein, if said selected character table output in step (e) is determined to be empty, said outputting step comprises outputting said current match.

8. The compression method of claim 6 wherein, if said selected character output in step (e) is determined to be empty, said inserting step includes

providing a next available string code, and

storing said next available string code in the location corresponding to said current match in said character table corresponding to said current character,

thereby inserting said extended string into said character table corresponding to said current character.

9. The compression method of claim 7 wherein said compression method operates in compression cycles, further including

setting said current match to said current character in preparation for performing a next compression cycle.

10. The compression method of claim 6 wherein said determining step comprises

including, at each location of said plurality of character tables, an empty flag representative of the empty status of said each location,

selecting the empty flag at the location corresponding to current match in the character table corresponding to current character, and

determining if said selected empty flag is representative of the empty status.

11. The compression method of claim 1 further comprising

creating a particular character table when the particular character corresponding to said particular character table is first encountered.

12. The compression method of claim 1 wherein a character table includes a list of prefix codes corresponding to the respective locations of said character table.

13. The compression method of claim 12 wherein said searching step includes

(a) matching one of said stored strings thereby providing a current match,

(b) fetching the next data character from said input stream following said current match thereby providing a current character,

(c) determining if said current match is one of the prefix codes in the prefix code list of said character table corresponding to said current character, and

(d) if said current match is one of said prefix codes, setting said current match to the code stored in the character table location corresponding to said one of said prefix codes and repeating steps (b) through (d) until said current match of step (c) is determined not to be one of said prefix codes in said prefix code list,

thereby determining said longest match.

14. The compression method of claim 13 wherein, if in step (c) said current match is determined not to be one of said prefix codes in said prefix code list,

said outputting step comprises outputting said current match.

15. The compression method of claim 13 wherein, if in step (c) said current match is determined not to be one of said prefix codes in said prefix code list, said inserting step includes

storing said current match in said prefix code list of said character table corresponding to said current character, thereby establishing a location corresponding to said current match in said character table corresponding to said current character,

providing a next available string code, and

storing said next available string code in said established location,

thereby inserting said extended string into said character table corresponding to said current character.

16. The compression method of claim 14 wherein said compression method operates in compression cycles, further including

setting said current match to said current character in preparation for performing a next compression cycle.

17. The compression method of claim 1 further comprising

establishing predetermined ones of said character tables in cache memory.

18. The compression method of claim 1 further comprising

establishing character tables corresponding to frequently occurring characters of said alphabet in cache memory.

19. The compression method of claim 1 further comprising

limiting strings stored in said character tables to a predetermined number of characters less than a predetermined string length limit.

20. The compression method of claim 11 further comprising

excluding creation of character tables corresponding to infrequently occurring characters of said alphabet.

21. The compression method of claim 12 further comprising

limiting the lengths of predetermined character tables to predetermined character table length limits, respectively.

22. Data compression apparatus for compressing an input stream of data characters into an output stream of compressed codes, said data characters being from an alphabet of data characters, comprising

a plurality of character tables corresponding to respective characters of said alphabet for storing strings of data characters encountered in said input stream, said stored strings having respective codes associated therewith, a string comprising a prefix string of at least one of said characters followed by an extension character,

a particular string being stored in said character tables by storing the code associated with said particular string in the character table corresponding to the extension character of said particular string at a character table location corresponding to the code of the prefix string of said particular string,

means for searching said input stream by comparing said input stream to said stored strings to determine the longest match therewith,

means for outputting the code associated with said longest match so as to provide said output stream of compressed codes, and

means for inserting an extended string into said character tables, said extended string comprising said longest match extended by the next data character in said input stream following said longest match, said extended string being stored in the character table corresponding to said next data character.

23. The compression apparatus of claim 22 wherein said searching means comprises means operative for

(a) matching one of said stored strings thereby providing a current match,

(b) fetching the next data character from said input stream following said current match thereby providing a current character,

(c) determining if the location corresponding to said current match in a character table corresponding to said current character is empty, and

(d) if said location is not empty, setting said current match to the code stored in said location and repeating (b) through (d) until the location of (c) is determined to be empty, thereby determining said longest match.

24. The compression apparatus of claim 23 wherein said outputting means comprises means for outputting said current match.

25. The compression apparatus of claim 23 wherein said inserting means includes

means for providing a next available string code, and

means for storing said next available string code in said empty location,

thereby inserting said extended string into said character table corresponding to said current character.

26. The compression apparatus of claim 24 wherein said compression apparatus operates in compression cycles, further including

means for setting said current match to said current character in preparation for performing a next compression cycle.

27. The compression apparatus of claim 22 wherein said searching means comprises means operative for

(a) matching one of said stored strings thereby providing a current match,

(b) fetching the next data character from said input stream following said current match thereby providing a current character,

(c) accessing the locations corresponding to said current match in said plurality of character tables so as to provide a plurality of character table outputs corresponding to said accessed locations, respectively,

(d) selecting the character table output from said character table corresponding to said current character,

(e) determining if said selected character table output is empty, and

(f) if said selected character table output is not empty, setting said current match to the code stored in said selected character table output and repeating (b) through (f) until the selected character table output of (e) is determined to be empty, thereby determining said longest match.

28. The compression apparatus of claim 27 wherein said outputting means comprises means for outputting said current match.

29. The compression apparatus of claim 27 wherein said inserting means includes

means for providing a next available string code, and

means for storing said next available string code in the location corresponding to said current match in said character table corresponding to said current character,

thereby inserting said extended string into said character table corresponding to said current character.

30. The compression apparatus of claim 28 wherein said compression apparatus operates in compression cycles, further including

means for setting said current match to said current character in preparation for performing a next compression cycle.

31. The compression apparatus of claim 27 further comprising an empty flag included at each location of said plurality of character tables representative of the empty status of said each location,

said searching means being operative for determining if said selected character table output is empty by selecting the empty flag at the location corresponding to current match in the character table corresponding to current character and determining if said selected empty flag is representative of the empty status.

32. The compression apparatus of claim 22 further comprising

means for creating a particular character table when the particular character corresponding to said particular character table is first encountered.

33. The compression apparatus of claim 22 wherein a character table includes a list of prefix codes corresponding to the respective locations of said character table.

34. The compression apparatus of claim 33 wherein said searching means comprises means operative for

(a) matching one of said stored strings thereby providing a current match,

(b) fetching the next data character from said input stream following said current match thereby providing a current character,

(c) determining if said current match is one of the prefix codes in the prefix code list of said character table corresponding to said current character, and

(d) if said current match is one of said prefix codes, setting said current match to the code stored in the character table location corresponding to said one of said prefix codes and repeating (b) through (d) until said current match of (c) is determined not to be one of said prefix codes in said prefix code list,

thereby determining said longest match.

35. The compression apparatus of claim 34 wherein said outputting means comprises means for outputting said current match.

36. The compression apparatus of claim 34 wherein said inserting means includes

means for storing said current match in said prefix code list of said character table corresponding to said current character, thereby establishing a location corresponding to said current match in said character table corresponding to said current character,

means for providing a next available string code, and

means for storing said next available string code in said established location,

thereby inserting said extended string into said character table corresponding to said current character.

37. The compression apparatus of claim 35 wherein said compression apparatus operates in compression cycles, further including

means for setting said current match to said current character in preparation for performing a next compression cycle.

38. The compression apparatus of claim 22 further comprising cache memory for storing predetermined ones of said character tables.

39. The compression apparatus of claim 22 further comprising cache memory for storing character tables corresponding to frequently occurring characters of said alphabet.

40. The compression apparatus of claim 22 further comprising

means for limiting strings stored in said character tables to a predetermined number of characters less than a predetermined string length limit.

41. The compression apparatus of claim 32 further comprising

means for excluding creation of character tables corresponding to infrequently occurring characters of said alphabet.

42. The compression apparatus of claim 33 further comprising

means for limiting the lengths of predetermined character tables to predetermined character table length limits, respectively.

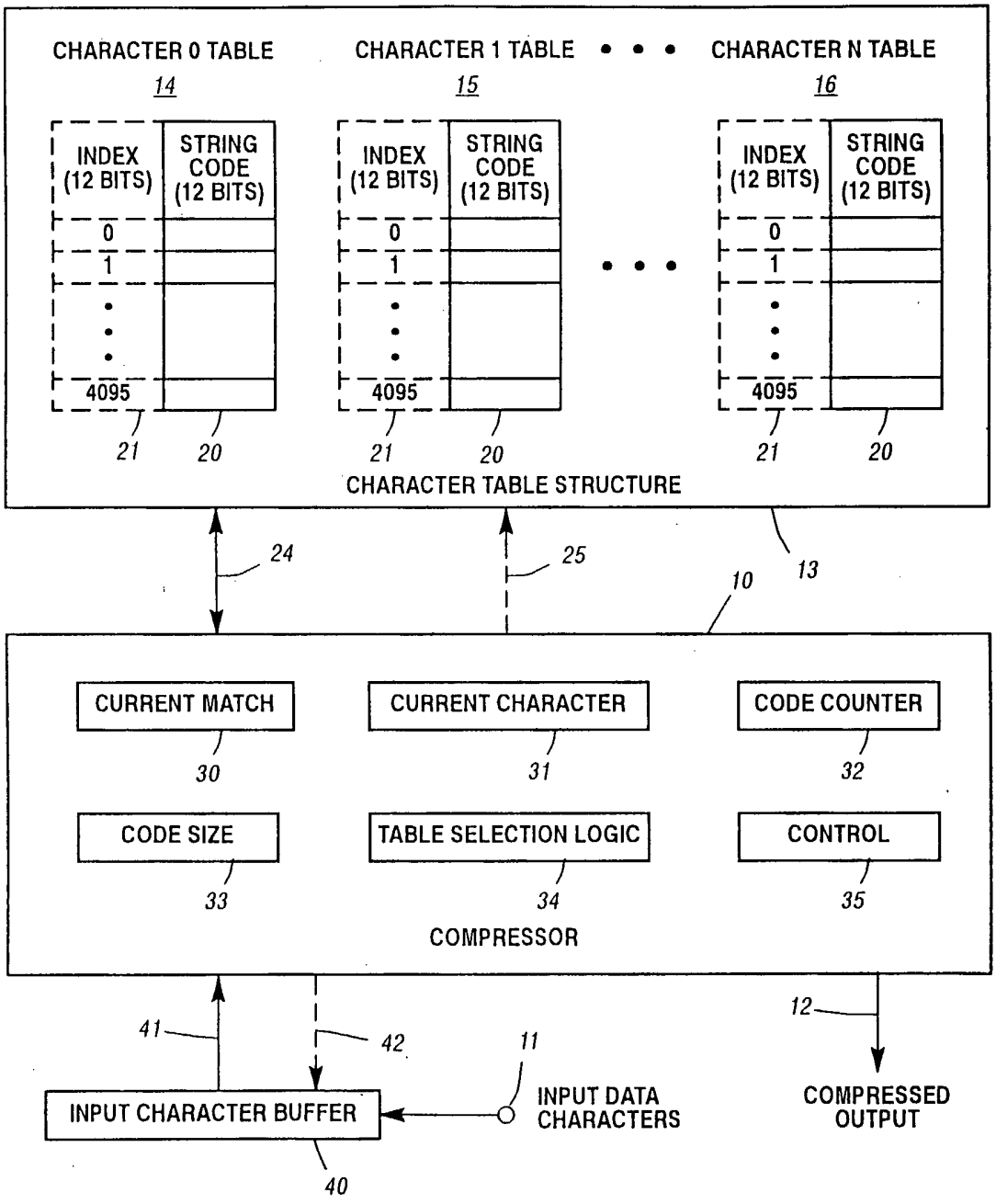

CHARACTER 0 TABLE
14
CHARACTER 1 TABLE
15
CHARACTER N TABLE
16
INDEX (12 BITS)
STRING CODE (12 BITS)
0
1
4095
21
20
CHARACTER TABLE STRUCTURE
24
25
13
10
CURRENT MATCH
30
CURRENT CHARACTER
31
CODE COUNTER
32
CODE SIZE
33
TABLE SELECTION LOGIC
34
CONTROL
35
COMPRESSOR
41
42
11
12
INPUT CHARACTER BUFFER
40
INPUT DATA CHARACTERS
COMPRESSED OUTPUT

Figure 1

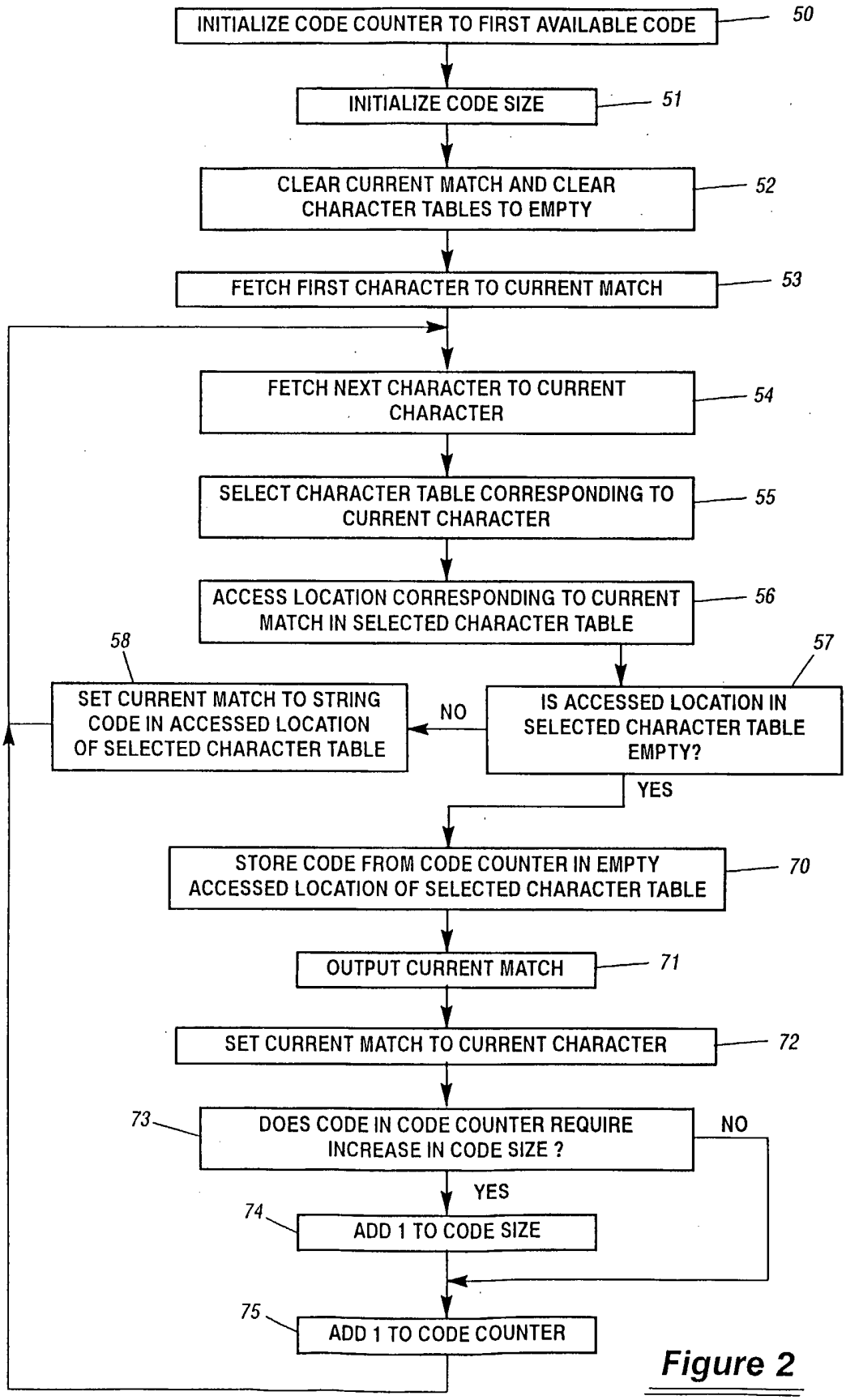

INITIALIZE CODE COUNTER TO FIRST AVAILABLE CODE
50
INITIALIZE CODE SIZE
51
CLEAR CURRENT MATCH AND CLEAR CHARACTER TABLES TO EMPTY
52
FETCH FIRST CHARACTER TO CURRENT MATCH
53
FETCH NEXT CHARACTER TO CURRENT CHARACTER
54
SELECT CHARACTER TABLE CORRESPONDING TO CURRENT CHARACTER
55
ACCESS LOCATION CORRESPONDING TO CURRENT MATCH IN SELECTED CHARACTER TABLE
56
58
57
SET CURRENT MATCH TO STRING CODE IN ACCESSED LOCATION OF SELECTED CHARACTER TABLE
NO
IS ACCESSED LOCATION IN SELECTED CHARACTER TABLE EMPTY?
YES
STORE CODE FROM CODE COUNTER IN EMPTY ACCESSED LOCATION OF SELECTED CHARACTER TABLE
70
OUTPUT CURRENT MATCH
71
SET CURRENT MATCH TO CURRENT CHARACTER
72
73
DOES CODE IN CODE COUNTER REQUIRE INCREASE IN CODE SIZE ?
NO
YES
74
ADD 1 TO CODE SIZE
75
ADD 1 TO CODE COUNTER

Figure 2

INPUT DATA CHARACTER STREAM

a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4 c_1 a_5 b_5 c_2 a_6 b_6 c_3 a_7 b_7 c_4 d_1

ACTION	CURR MATCH	CURR CHAR	CODE CNTR	CHAR TABLE	INDEX (PREFIX)	STRING CODE	OUTPUT	BLOCK OF FIG. 2
1	a_1	b_1	258	b	a	258	a_1	50-57,70,71
2	b_1	a_2	259	a	b	259	b_1	72-75,54-57,70,71
3	a_2	b_2	260	b	a	(258)		72-75,54-57
4	258	a_3		a	258	260	258	58,54-57,70,71
5	a_3	b_3	261	b	a	(258)		72-75,54-57
6	258	a_4		a	258	(260)		58,54-57
7	260	b_4		b	260	261	260	58,54-57,70,71
8	b_4	c_1	262	c	b	262	b_4	72-75,54-57,70,71
9	c_1	a_5	263	a	c	263	c_1	72-75,54-57,70,71
10	a_5	b_5	264	b	a	(258)		72-75,54-57
11	258	c_2		c	258	264	258	58,54-57,70,71
12	c_2	a_6	265	a	c	(263)		72-75,54-57
13	263	b_6		b	263	265	263	58,54-57,70,71
14	b_6	c_3	266	c	b	(262)		72-75,54-57
15	262	a_7		a	262	266	262	58,54-57,70,71
16	a_7	b_7	267	b	a	(258)		72-75,54-57
17	258	c_4		c	258	(264)		58,54-57
18	264	d_1		d	264	267	264	58,54-57,70,71
19	d_1	..	268	.	...	...	d_1	72-75,54,71,END

Figure 3

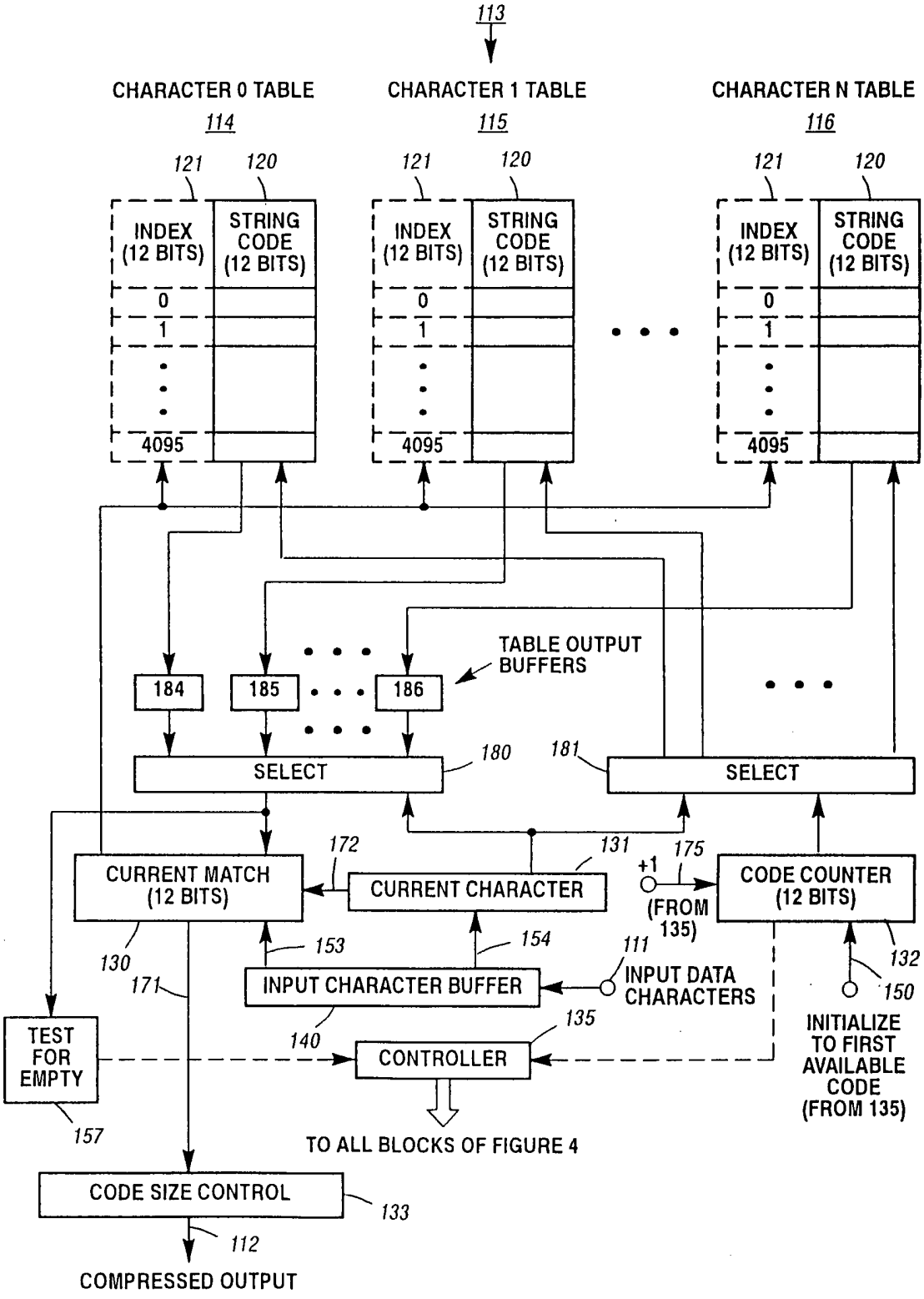

113
CHARACTER 0 TABLE
114
CHARACTER 1 TABLE
115
CHARACTER N TABLE
116
121
120
INDEX (12 BITS)
STRING CODE (12 BITS)
0
1
4095
TABLE OUTPUT BUFFERS
184
185
186
SELECT
180
181
SELECT
172
131
175
CURRENT MATCH (12 BITS)
CURRENT CHARACTER
+1
(FROM 135)
CODE COUNTER (12 BITS)
130
153
154
111
132
171
INPUT CHARACTER BUFFER
INPUT DATA CHARACTERS
150
INITIALIZE TO FIRST AVAILABLE CODE (FROM 135)
TEST FOR EMPTY
140
135
CONTROLLER
157
TO ALL BLOCKS OF FIGURE 4
CODE SIZE CONTROL
133
112
COMPRESSED OUTPUT

Figure 4

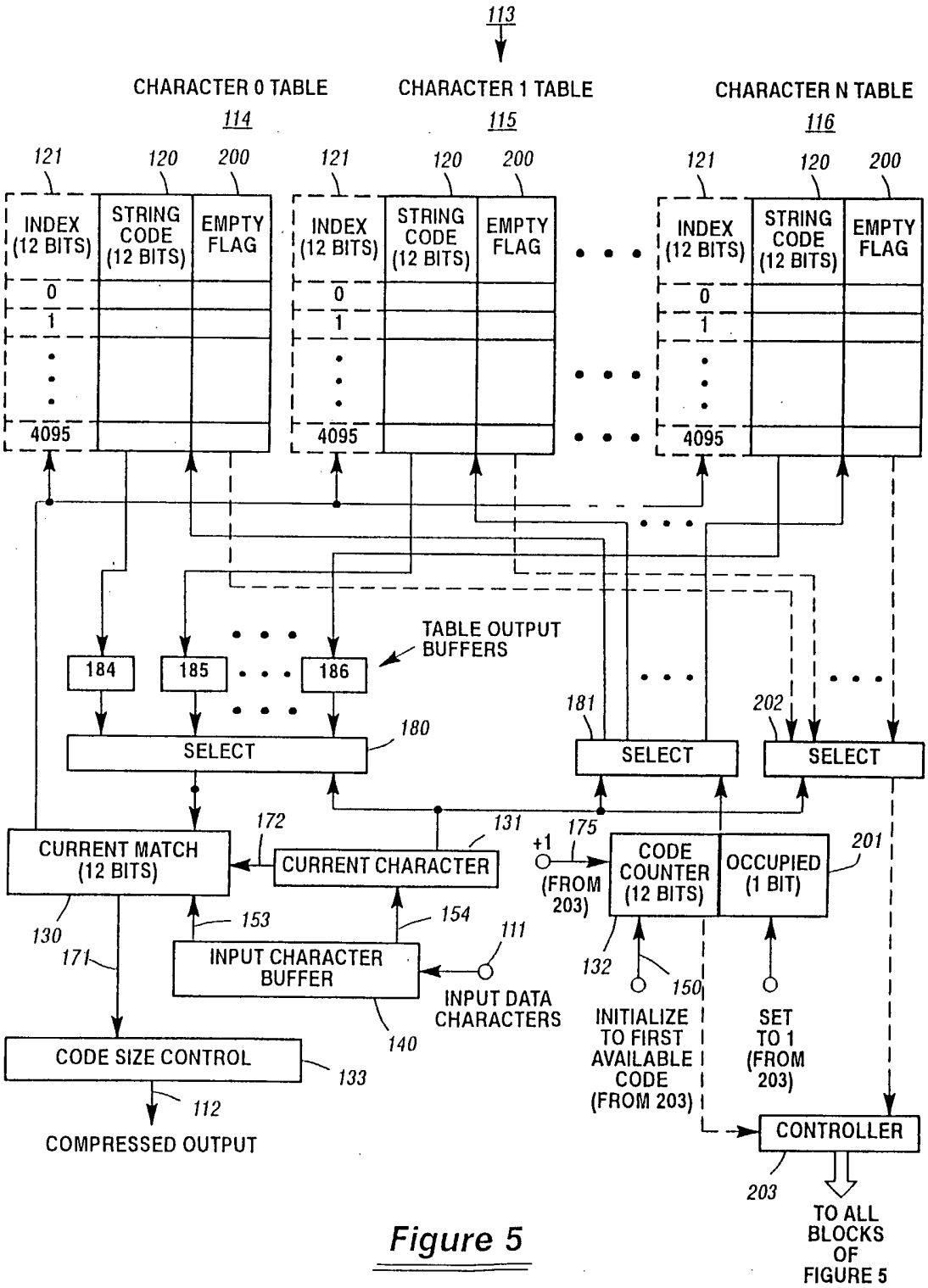

113
CHARACTER 0 TABLE
114
CHARACTER 1 TABLE
115
CHARACTER N TABLE
116
121
120
200
INDEX (12 BITS)
STRING CODE (12 BITS)
EMPTY FLAG
0
1
4095
TABLE OUTPUT BUFFERS
184
185
186
180
181
202
SELECT
172
131
175
CURRENT MATCH (12 BITS)
CURRENT CHARACTER
+1
(FROM 203)
CODE COUNTER (12 BITS)
OCCUPIED (1 BIT)
201
130
153
154
111
171
INPUT CHARACTER BUFFER
INPUT DATA CHARACTERS
132
150
INITIALIZE TO FIRST AVAILABLE CODE (FROM 203)
SET TO 1 (FROM 203)
140
CODE SIZE CONTROL
133
112
COMPRESSED OUTPUT
CONTROLLER
203
TO ALL BLOCKS OF FIGURE 5

Figure 5

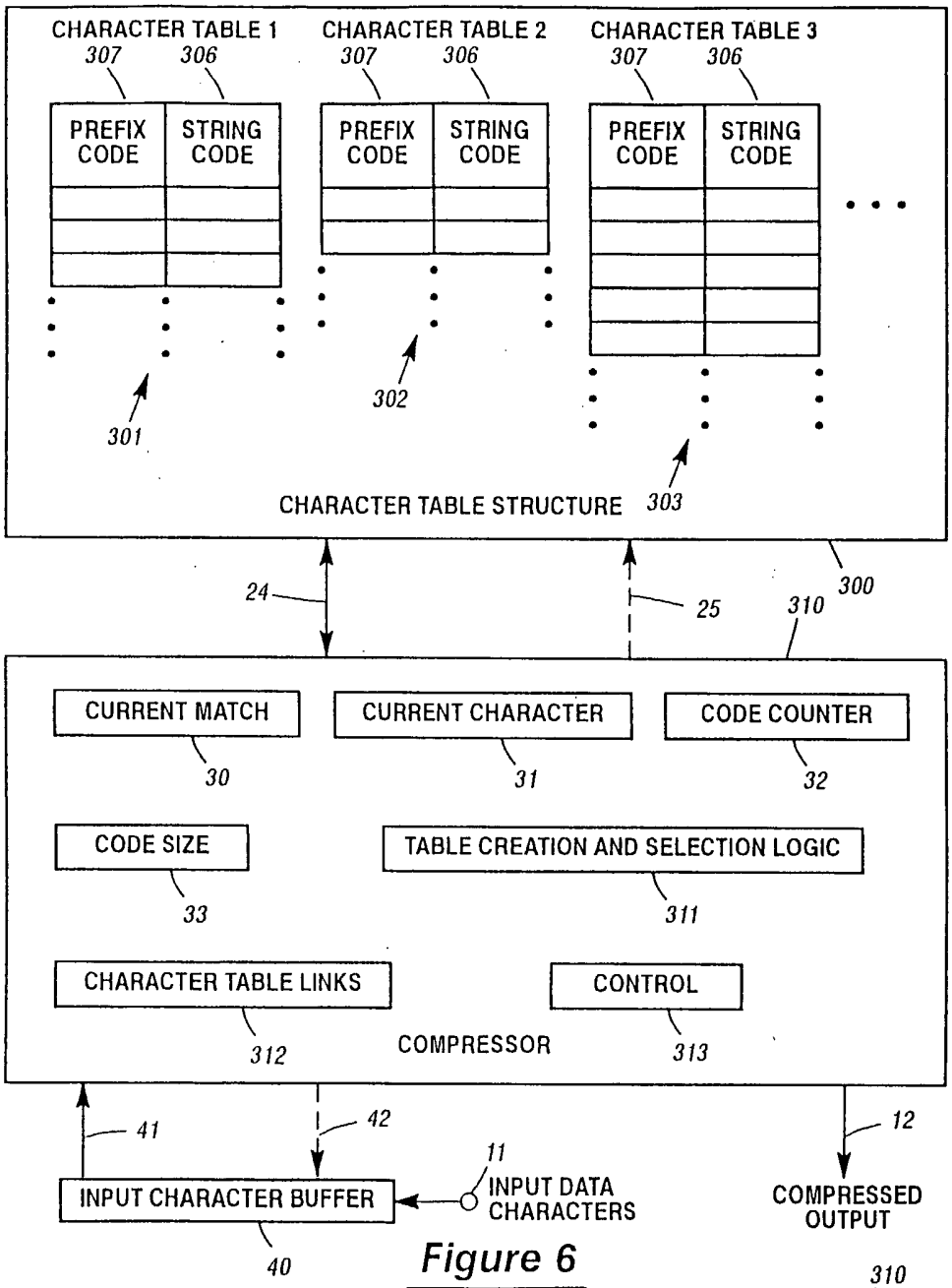


Figure 6

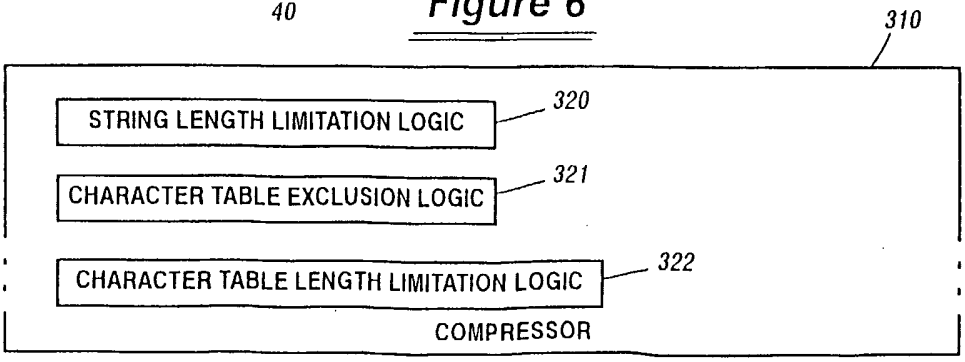


Figure 9

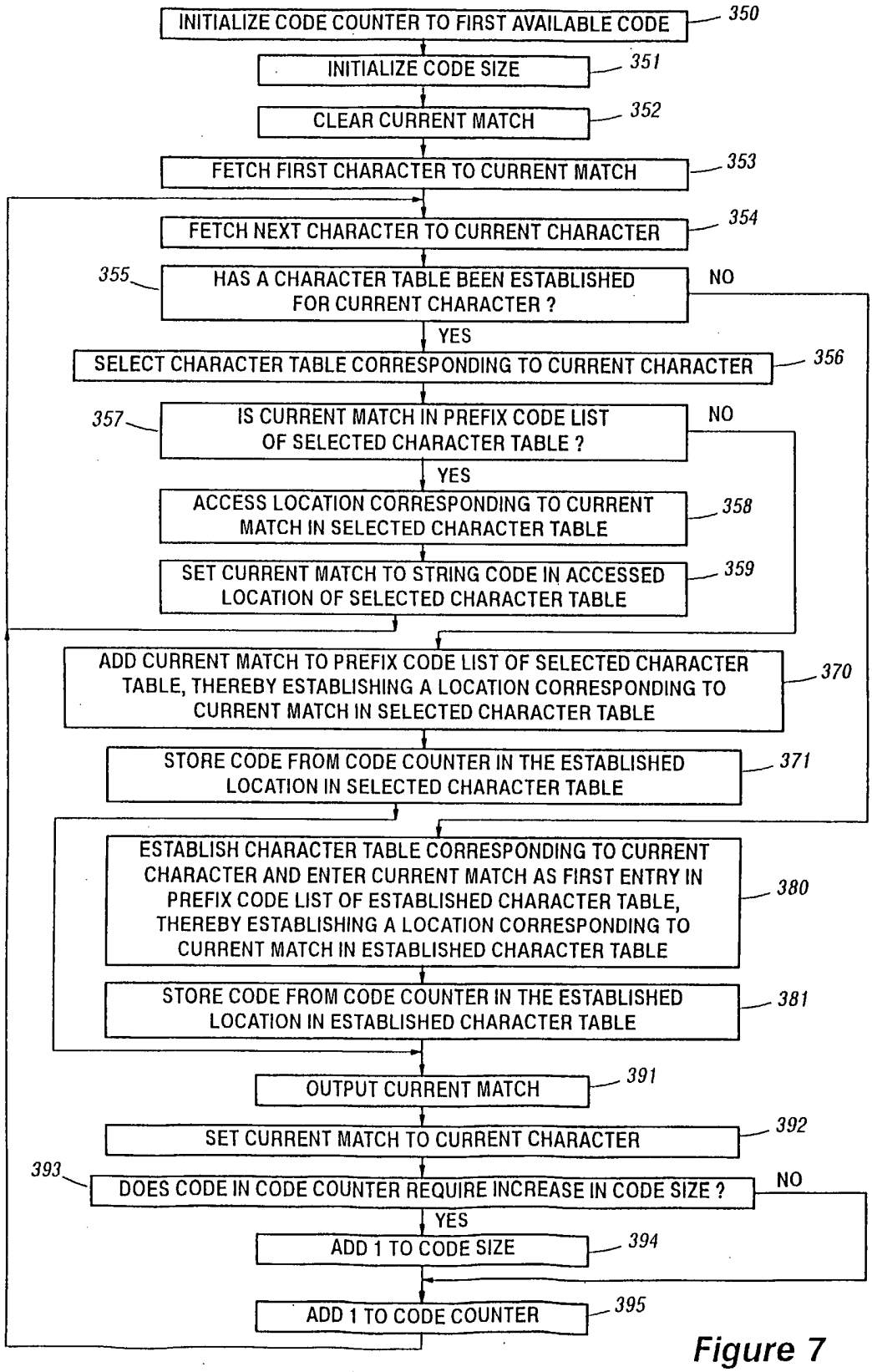

INITIALIZE CODE COUNTER TO FIRST AVAILABLE CODE
350
INITIALIZE CODE SIZE
351
CLEAR CURRENT MATCH
352
FETCH FIRST CHARACTER TO CURRENT MATCH
353
FETCH NEXT CHARACTER TO CURRENT CHARACTER
354
355
HAS A CHARACTER TABLE BEEN ESTABLISHED FOR CURRENT CHARACTER ?
NO
YES
SELECT CHARACTER TABLE CORRESPONDING TO CURRENT CHARACTER
356
357
IS CURRENT MATCH IN PREFIX CODE LIST OF SELECTED CHARACTER TABLE ?
NO
YES
ACCESS LOCATION CORRESPONDING TO CURRENT MATCH IN SELECTED CHARACTER TABLE
358
SET CURRENT MATCH TO STRING CODE IN ACCESSED LOCATION OF SELECTED CHARACTER TABLE
359
ADD CURRENT MATCH TO PREFIX CODE LIST OF SELECTED CHARACTER TABLE, THEREBY ESTABLISHING A LOCATION CORRESPONDING TO CURRENT MATCH IN SELECTED CHARACTER TABLE
370
STORE CODE FROM CODE COUNTER IN THE ESTABLISHED LOCATION IN SELECTED CHARACTER TABLE
371
ESTABLISH CHARACTER TABLE CORRESPONDING TO CURRENT CHARACTER AND ENTER CURRENT MATCH AS FIRST ENTRY IN PREFIX CODE LIST OF ESTABLISHED CHARACTER TABLE, THEREBY ESTABLISHING A LOCATION CORRESPONDING TO CURRENT MATCH IN ESTABLISHED CHARACTER TABLE
380
STORE CODE FROM CODE COUNTER IN THE ESTABLISHED LOCATION IN ESTABLISHED CHARACTER TABLE
381
OUTPUT CURRENT MATCH
391
SET CURRENT MATCH TO CURRENT CHARACTER
392
393
DOES CODE IN CODE COUNTER REQUIRE INCREASE IN CODE SIZE ?
NO
YES
ADD 1 TO CODE SIZE
394
ADD 1 TO CODE COUNTER
395

Figure 7

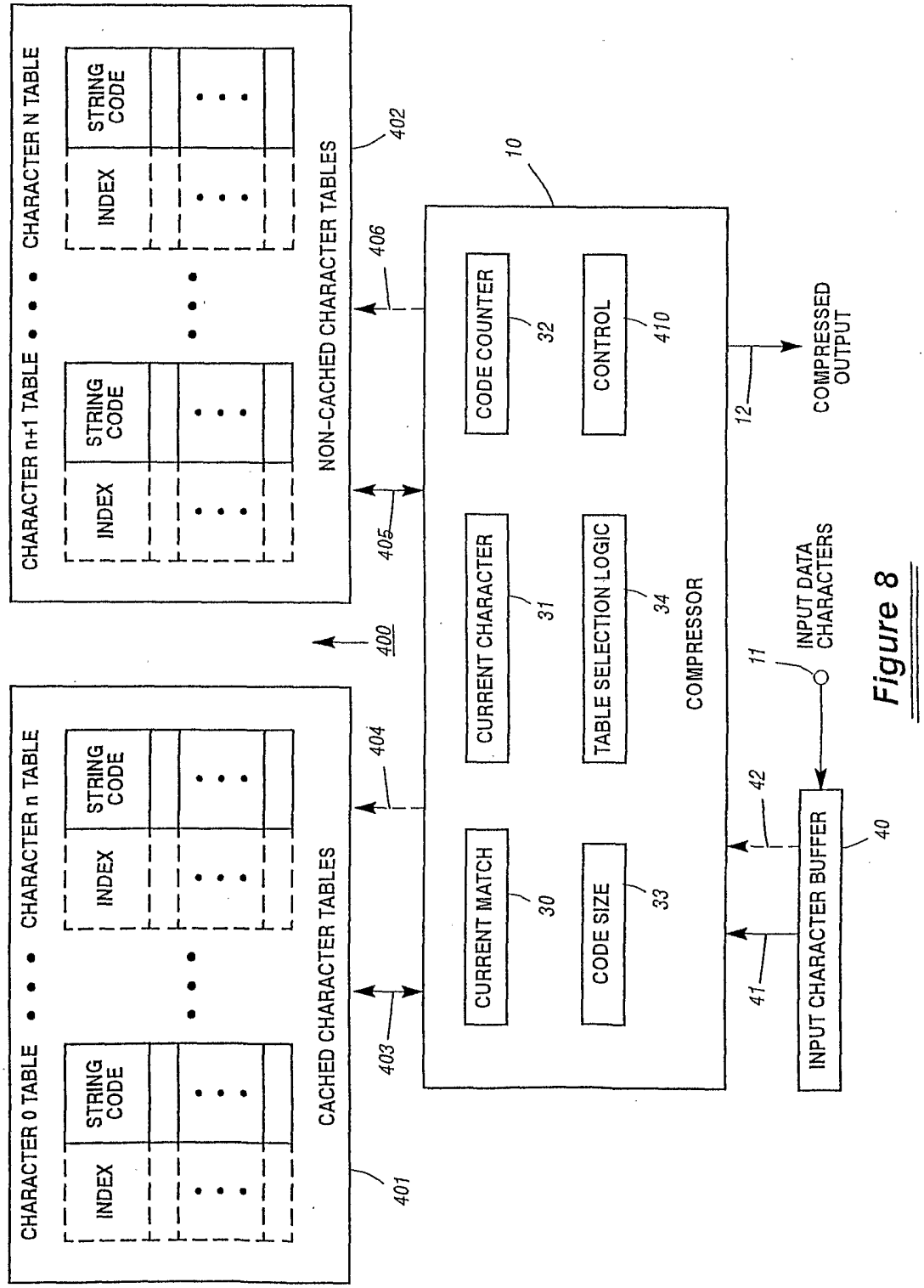

CHARACTER 0 TABLE
CHARACTER n TABLE
INDEX
STRING CODE
CACHED CHARACTER TABLES
401
403
404
CHARACTER n+1 TABLE
CHARACTER N TABLE
NON-CACHED CHARACTER TABLES
402
406
405
400
10
CURRENT MATCH
30
CURRENT CHARACTER
31
CODE COUNTER
32
CODE SIZE
33
TABLE SELECTION LOGIC
34
CONTROL
410
COMPRESSOR
12
COMPRESSED OUTPUT
11
INPUT DATA CHARACTERS
41
42
INPUT CHARACTER BUFFER
40

Figure 8

INTERNATIONAL SEARCH REPORT

Int ial Application No
PCT/US 02/11989

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H03M7/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H03M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 818 873 A (WINTERS KEL D ET AL) 6 October 1998 (1998-10-06) abstract column 5, line 60 -column 6, line 44	1-42
A	US 6 169 499 B1 (COOPER ALBERT B) 2 January 2001 (2001-01-02) the whole document	1-42
A	US 5 455 576 A (CLARK II AIRELL R ET AL) 3 October 1995 (1995-10-03) abstract; claims	1-42
A	US 5 838 264 A (COOPER ALBERT B) 17 November 1998 (1998-11-17) the whole document	1-42

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 September 2002	07/10/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Foglia, P

INTERNATIONAL SEARCH REPORT

Inte[illegible]nal Application No
PCT/US 02/11989

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5818873 A	06-10-1998	JP 7509823 T	26-10-1995
		WO 9403983 A1	17-02-1994
US 6169499 B1	02-01-2001	AU 5874900 A	09-01-2001
		EP 1195010 A1	10-04-2002
		WO 0079687 A1	28-12-2000
US 5455576 A	03-10-1995	US 5373290 A	13-12-1994
		DE 69518022 D1	24-08-2000
		DE 69518022 T2	21-12-2000
		EP 0666651 A2	09-08-1995
		JP 3309028 B2	29-07-2002
		JP 7273667 A	20-10-1995
		DE 69318064 D1	28-05-1998
		DE 69318064 T2	13-08-1998
		EP 0573208 A1	08-12-1993
		HK 1011122 A1	24-03-2000
US 5838264 A	17-11-1998	US 5642112 A	24-06-1997
		AT 178442 T	15-04-1999
		AU 702620 B2	25-02-1999
		AU 4525996 A	24-07-1996
		CA 2208049 A1	11-07-1996
		CN 1171868 A	28-01-1998
		DE 69508796 D1	06-05-1999
		DE 69508796 T2	19-08-1999
		DK 800726 T3	11-10-1999
		EP 0800726 A1	15-10-1997
		ES 2130696 T3	01-07-1999
		FI 972779 A	27-06-1997
		GR 3030107 T3	30-07-1999
		JP 3016868 B2	06-03-2000
		JP 10508170 T	04-08-1998
		RU 2159989 C2	27-11-2000
		WO 9621283 A1	11-07-1996