PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 5/00	A1	(11) International Publication Number: WO 00/04442
		(43) International Publication Date: 27 January 2000 (27.01.00)

(21) International Application Number: PCT/US99/13293

(22) International Filing Date: 11 June 1999 (11.06.99)

(30) Priority Data:
09/115,445 14 July 1998 (14.07.98) US

(71) Applicant: MOTOROLA INC. [US/US]; 1303 East Algonquin Road, Schaumburg, IL 60196 (US).

(72) Inventors: WANG, Zhonghe; 61 Cannon Road, Lake Worth, FL 33467 (US). CHANG, Ying–Yueh; 672 N.W. 109th Terrace, Coral Springs, FL 33071 (US). SCHWENDEMAN, Robert, John; 590 S.E. 10th Avenue, Pompano Beach, FL 33060 (US).

(74) Agents: NICHOLS, Daniel, K. et al.; Motorola Inc., Intellectual Property Department, 1500 Gateway Boulevard, Boynton Beach, FL 33426–8292 (US).

(81) Designated States: AU, BR, CA, MX, ZA, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: REDUCED OVERHEAD TEXT MESSAGING

(57) Abstract

A reduced overhead text messaging system (900) has a lossless compression engine (1104) that operates on an original message to generate a compressed message. The compressed message includes at least one of a data type switch token and a fragment and phrase token. The compressed message is received by a messaging unit (906) and decompressed for presentation.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

REDUCED OVERHEAD TEXT MESSAGING

Field of the Invention

This invention relates in general to selective call signaling systems and more particularly to a selective call signaling system that facilitates reduced overhead text messaging over a wireless network.

Background of the Invention

In conventional selective call signaling systems, a user or originator may send a message to a messaging unit (e.g., selective call receiver), the message comprising an address associated with the messaging unit, and data. The data may be in one or more forms such as numeric digits representing a phone number, alphanumeric characters representing a readable text message, or possibly a multimedia message comprising audio and graphical information. Typically, in its conventional form, this type of messaging is sufficient to convey information between individuals or services relating to their business, special interests, whereabouts, general scheduling, or time critical appointments. However, because of an increasing need to timely communicate large amounts of information between subscribers, a solution must be found that reduces the overall amount of data on a given channel, thus allowing higher data throughput, increased channel utilization, and reduced messaging latency.

Accordingly, what is needed is wireless messaging system that allows an originator to communicate a text message to a messaging unit in a form that reduces the total overhead associated with text messaging.

Summary of the Invention

Briefly, according to the invention, there is provided a method and apparatus for sending data comprising compressed text messages, using existing selective call signaling equipment operating with paging protocols such as FLEX™, a trademark of Motorola, Inc., POCSAG (Post Office Code Standardisation Advisory Group), or the like.

A first aspect of the invention involves realizing hardware that implements a method for overlaying compressed text messaging on an existing paging infrastructure. The existing paging infrastructure comprises a paging terminal that includes a paging encoder for processing received messages containing original text messages and their corresponding destination requests, e.g., selective call Personal Identification Numbers (PIN's), cap codes, selective call addresses, or the like. The paging terminal generates a messaging queue of compressed selective call messages comprising the received messages and their corresponding selective call address(es), as determined from the corresponding destination requests. Distribution of the selective call messages in the

messaging queue is handled by the paging terminal which dispatches messages to at least one base station (e.g., transmitter, antenna, and receiver) for communication between the base station and the messaging unit(s) or pagers.

A second aspect of the invention involves the inclusion of a lossless compression engine, preferably in the paging terminal, for selectively compressing messages received from the originator or a two-way messaging unit, e.g. portable data terminal, cellular telephone, or two-way pager.

A third aspect of the invention involves the messaging unit or pager that is equipped with a lossless decompression engine that can decompress compressed messaging information contained in the compressed selective call messages to recover the original text.

A fourth aspect of the invention involves the messaging unit or pager being equipped with a primary and possibly a secondary apparatus for communicating both inbound and outbound messages. The primary apparatus comprises a conventional radio frequency receiver and optionally a conventional radio frequency transmitter. The secondary apparatus comprises an optical receiver and optionally an optical transmitter. Alternatively, the secondary apparatus may further comprise one or more acoustic or other electromagnetic transducers and associated circuitry implementing a uni- or bi-directional communication link between the messaging unit or pager and the originator.

A fifth aspect of the invention involves the compression engine associated with the paging terminal and the decompression engine associated with the messaging unit or pager accommodating a plurality of compression procedures. These compression procedures comprise both lossless or lossy compression schemes, as appropriate for the information being compressed.

A sixth aspect of the invention involves the messaging unit or pager including a compression engine for compressing a message newly originating from the messaging unit or pager and destined for at least one messaging unit or pager having message decompression capability.

Brief Description of the Drawings

FIG. 1 is an electrical block diagram of a data transmission system for use in accordance with the preferred embodiment of the present invention.

FIG. 2 is an electrical block diagram of a terminal for processing and transmitting message information in accordance with the preferred embodiment of the present invention.

FIGs. 3-5 are timing diagrams illustrating the transmission format of the signaling protocol utilized in accordance with the preferred embodiment of the present invention.

FIGs. 6 and 7 are timing diagrams illustrating the synchronization signals utilized in accordance with the preferred embodiment of the present invention.

FIG. 8 is an electrical block diagram of a messaging unit in accordance with the preferred embodiment of the present invention.

FIG. 9 is a diagram of a compressed messaging system in accordance with the present invention.

FIG. 10 is a high level block diagram of a messaging unit in accordance with the preferred embodiment of the present invention.

FIG. 11 is a block diagram of the message composition and compression equipment that could be used to send compressed messages to subscriber messaging units via a paging channel.

FIG. 12 is a functional diagram of a wireless selective call signaling system controller that implements a combined one-way and two-way compressed messaging system capable of communicating with the messaging units.

FIG. 13 is an exemplary state diagram showing the stages for a string of length n.

FIG. 14 is a state table depicting a typical parsing and compression operation of the compression engine to resolve the shortest path for optimal text compression in accordance with the preferred embodiment of the present invention.

FIG. 15 is an exemplary state diagram showing a typical initial data type assignment in accordance with the preferred embodiment of the present invention.

FIG. 16 is an example of shortest path encoding of a compressed message including details such as message segmentation, shortest path data encoding, shortest path data type switch encoding, and padding in accordance with the preferred embodiment of the present invention.

Description of a Preferred Embodiment

Referring to FIG. 1, an electrical block diagram illustrates a data transmission system **100**, such as a paging system, for use in accordance with the preferred embodiment of the present invention. In the data transmission system **100**, messages originating either from a phone, as in a system providing numeric data transmission, or from a message entry device, such as an alphanumeric data terminal, are routed through the public switched telephone network (PSTN) to a paging terminal **102** which processes the numeric or alphanumeric message information for transmission by one or more transmitters **104** provided within the system. When multiple transmitters are utilized, the transmitters **104** may use simulcast, multicast, or zone based transmitting schemes to broadcast the message information to messaging units **106**. Processing of the numeric and alphanumeric information by the paging terminal **102**, and the protocol utilized for the transmission of the messages is described in detail in the following text.

Referring to FIG. 2, an electrical block diagram illustrates the paging terminal **102** utilized for processing and controlling the transmission of the message information in accordance with the preferred embodiment of the present invention. Short messages, such as tone-only and numeric messages which can be readily entered using a conventional dual tone multi-frequency telephone are coupled to the paging terminal **102** through a telephone interface **202** in a manner well known in the art. Longer messages, such as alphanumeric messages which require the use of a data entry device are coupled to the paging terminal **102** through a modem **206** or the like using any of a

number of well known transmission protocol standards. When a call to place a message is received, a controller **204** handles the processing of the message. The controller **204** is preferably a microcomputer, such as manufactured by Motorola Inc., and which runs various pre-programmed routines for controlling such terminal operations as voice prompts to direct the caller to enter the message, or the handshaking protocol to enable reception of messages from a data entry device. When a call is received, the controller **204** references information stored in the subscriber database **208** to determine how the message being received is to be processed. The subscriber data base **208** includes, but is not limited to such information as addresses assigned to the messaging unit, message type associated with the address, and information related to the status of the messaging unit, such as active or inactive for failure to pay the bill. A data entry terminal **240** is provided which couples to the controller **204**, and which is used for such purposes as entry, updating and deleting of information stored in the subscriber data base **208**, for monitoring system performance, and for obtaining such information as billing information.

The subscriber database **208** also includes such information as to what transmission frame and to what transmission phase the messaging unit is assigned, as will be described in further detail below. The received message is stored in an active page file **210** which stores the messages in queues according to the transmission phase assigned to the messaging unit. In the preferred embodiment of the present invention, four phase queues are provided in the active page file **210**. The active page file **210** is preferably a dual port, first in first out random access memory, although it will be appreciated that other random access memory devices, such as hard disk drives, can be utilized as well. Periodically the message information stored in each of the phase queues is recovered from the active page file **210** under control of controller **204** using timing information such as provided by a real time clock **214**, or other suitable timing source. The recovered message information from each phase queue is sorted by frame number and is then organized by address, message information, and any other information required for transmission (all of which is referred to as message related information), and then batched into frames based upon message size by frame batching controller **212**. The batched frame information for each phase queue is coupled to frame message buffers **216** which temporarily store the batched frame information until a time for further processing and transmission. Frames are batched in numeric sequence, so that while a current frame is being transmitted, the next frame to be transmitted is in the frame message buffer **216**, and the next frame thereafter is being retrieved and batched. At the appropriate time, the batched frame information stored in the frame message buffer **216** is transferred to the frame encoder **218**, again maintaining the phase queue relationship. The frame encoder **218** encodes the address and message information into address and message codewords required for transmission, as will be described below. The encoded address and message codewords are ordered into blocks and then coupled to a block interleaver **220** which interleaves preferably eight codewords at a time to form interleaved information blocks for transmission in a manner well known in the art. The interleaved codewords contained in the interleaved information blocks produced by each block interleaver **220** are then serially transferred to a phase multiplexer **221**, which multiplexes the

message information on a bit by bit basis into a serial data stream by transmission phase. The controller **204** next enables a frame sync generator **222** which generates the synchronization code which is transmitted at the start of each frame transmission. The synchronization code is multiplexed with address and message information under the control of controller **204** by serial data splicer **224**, and generates therefrom a message stream which is properly formatted for transmission. The message stream is next coupled to a transmitter controller **226**, which under the control of controller **204** transmits the message stream over a distribution channel **228**. The distribution channel **228** may be any of a number of well known distribution channel types, such as wire line, an RF or microwave distribution channel, or a satellite distribution link. The distributed message stream is transferred to one or more transmitter stations **104**, depending upon the size of the communication system. The message stream is first transferred into a dual port buffer **230** which temporarily stores the message stream prior to transmission. At an appropriate time determined by timing and control circuit **232**, the message stream is recovered from the dual port buffer **230** and coupled to the input of preferably a 4-level FSK modulator **234**. The modulated message stream is then coupled to the transmitter **236** for transmission via antenna **238**.

Referring to FIGs. 3, 4 and 5, the timing diagrams illustrate the transmission format of the signaling protocol utilized in accordance with the preferred embodiment of the present invention. This signaling protocol is commonly referred to as Motorola's™ FLEX™ selective call signaling protocol. As shown in FIG. 3, the signaling protocol enables message transmission to messaging units, such as pagers, assigned to one or more of 128 frames which are labeled frame 0 through frame 127. It then will be appreciated that the actual number of frames provided within the signaling protocol can be greater or less than described above. The greater the number of frames utilized, the greater the battery life that may be provided to the messaging units operating within the system. The fewer the number of frames utilized, the more often messages can be queued and delivered to the messaging units assigned to any particular frame, thereby reducing the latency, or time required to deliver messages.

As shown in FIG. 4, the frames comprise a synchronization codeword (sync) followed preferably by eleven blocks of message information (information blocks) which are labeled block 0 through block 10. As shown in FIG. 5., each block of message information comprises preferably eight address, control or data codewords which are labeled word 0 through word 7 for each phase. Consequently, each phase in a frame allows the transmission of up to eighty-eight address, control and data codewords. The address, control and data codewords preferably comprise two sets, a set first relating to a vector field comprising a short address vector, a long address vector, a first message word, and a null word, and a second set relating to a message field comprising a message word and a null word.

The address, control, and data or message codewords are preferably 31,21 BCH codewords with an added thirty-second even parity bit which provides an extra bit of distance to the codeword set. It will be appreciated that other codewords, such as a 23,12 Golay codeword could be utilized as well. Unlike the well known POCSAG signaling protocol which provides

address and data codewords which utilize the first codeword bit to define the codeword type, as either address or data, no such distinction is provided for the address and data codewords in the FLEX™ signaling protocol utilized with the preferred embodiment of the present invention. Rather, address and data codewords are defined by their position within the individual frames.

FIGS. 6 and 7 are timing diagrams illustrating the synchronization code utilized in accordance with the preferred embodiment of the present invention. In particular, as shown in FIG. 6, the synchronization code comprises preferably three parts, a first synchronization code (sync 1), a frame information codeword (frame info) and a second synchronization codeword (sync 2). As shown in FIG. 7, the first synchronization codeword comprises first and third portions, labeled bit sync 1 and BS1, which are alternating 1,0 bit patterns which provides bit synchronization, and second and fourth portions, labeled "A" and its complement "A bar", which provide frame synchronization. The second and fourth portions are preferably single 32,21 BCH codewords which are predefined to provide high codeword correlation reliability, and which are also used to indicate the data bit rate at which addresses and messages are transmitted. Table 1 defines the data bit rates which are used in conjunction with the signaling protocol.

Bit Rate	**"A" Value**
1600 bps	A1 and A1 bar
3200 bps	A2 and A2 bar
6400 bps	A3 and A3 bar
Not defined	A4 and A4 bar

Table 1

As shown in Table 1, three data bit rates are predefined for address and message transmission, although it will be appreciated that more or less data bit rates can be predefined as well, depending upon the system requirements.

The frame information codeword is preferably a single 32,21 BCH codeword which includes within the data portion a predetermined number of bits reserved to identify the frame number, such as 7 bits encoded to define frame number 0 to frame number 127.

The structure of the second synchronization code is preferably similar to that of the first synchronization code described above. However, unlike the first synchronization code which is preferably transmitted at a fixed data symbol rate, such as 1600 bps (bits per second), the second synchronization code is transmitted at the data symbol rate at which the address and messages are to be transmitted in any given frame. Consequently, the second synchronization code allows the messaging unit to obtain "fine" bit and frame synchronization at the frame transmission data bit rate.

In summary the signaling protocol utilized with the preferred embodiment of the present invention comprises 128 frames which include a predetermined synchronization code followed by eleven information blocks which comprise eight address, control or message codewords per phase. The synchronization code enables identification of the data transmission rate, and insures synchronization by the messaging unit with the data codewords transmitted at the various transmission rates.

FIG. 8 is an electrical block diagram of the messaging unit **106** in accordance with the preferred embodiment of the present invention. The heart of the messaging unit **106** is a controller **816**, which is preferably implemented using a low power microcomputer, such as manufactured by Motorola, Inc., or the like. The microcomputer controller, hereinafter call the controller **816**, receives and processes inputs from a number of peripheral circuits, as shown in FIG. 8, and controls the operation and interaction of the peripheral circuits using software subroutines. The use of a microcomputer controller for processing and control functions (e.g., as a function controller) is well known to one of ordinary skill in the art.

The messaging unit **106** is capable of receiving address, control and message information, hereafter called "data" which is modulated using preferably 2-level and 4-level frequency modulation techniques. The transmitted data is intercepted by an antenna **802** which couples to the input of a receiver section **804**. Receiver section **804** processes the received data in a manner well known in the art, providing at the output an analog 4-level recovered data signal, hereafter called a recovered data signal. The recovered data signal is coupled to one input of a threshold level extraction circuit **808**, and to an input of a 4-level decoder **810**.

Operation of the threshold level extraction circuit **808**, 4-level decoder **810**, symbol synchronizer **812**, 4-level to binary converter **814**, synchronization codeword correlator **818**, and phase timing generator (data recovery timing circuit) **826** depicted in the messaging unit of FIG. 8 is best understood with reference to United States Patent No. 5,282,205 entitled "Data Communication Terminal Providing Variable Length Message Carry-On And Method Therefor,"

issued to Kuznicki et al., assigned to Motorola, Inc., the teachings of which are incorporated herein by reference thereto.

Again referring to FIG. 8, the threshold level extraction circuit **808** comprises two clocked level detector circuits (not shown) which have as inputs the recovered data signal. Preferably, signal states of 17%, 50% and 83%, are utilized to enable decoding the 4-level data signals presented to the threshold level extraction circuit **808**.

When power is initially applied to the receiver portion, as when the messaging unit is first turned on, a clock rate selector is preset through a control input (center sample) to select a 128X clock, i.e. a clock having a frequency equivalent to 128 times the slowest data bit rate, which as described above is 1600 bps. The 128X clock is generated by 128X clock generator **844**, as shown in FIG. 8, which is preferably a crystal controlled oscillator operating at 204.8 KHz (kilohertz). The output of the 128X clock generator **844** couples to an input of frequency divider **846** which divides the output frequency by two to generate a 64X clock at 102.4 KHz. The 128X clock allows the level detectors to asynchronously detect in a very short period of time the peak and valley signal amplitude values, and to therefore generate the low (Lo), average (Avg) and high (Hi) threshold output signal values required for modulation decoding. After symbol synchronization is achieved with the synchronization signal, as will be described below, the controller **816** generates a second control signal (center sample) to enable selection of a 1X symbol clock which is generated by symbol synchronizer **812** as shown in FIG. 8.

The 4-level decoder **810** preferably operates using three voltage comparators and a symbol decoder. The recovered data signal is coupled to an input of the three comparators having thresholds corresponding with normalized signal states of 17%, 50% and 83%. The resulting system effectively recovers the demodulated 2- or 4- level FSK information signal by coupling the recovered data signal to the second input of an 83% comparator, the second input of a 50% comparator, and the second input of a 17% comparator. The outputs of the three comparators corresponding with the low (Lo), average (Avg) and high (Hi) threshold output signal values are coupled to inputs of a symbol decoder. The symbol decoder then decodes the inputs according to Table 2.

Threshold			**Output**	
Hi	Avg	Lo	MSB	LSB
RC_{in} <	RC_{in} <	RC_{in} <	0	0
RC_{in} <	RC_{in} <	RC_{in} >	0	1
RC_{in} <	RC_{in} >	RC_{in} >	1	1
RC_{in} >	RC_{in} >	RC_{in} >	1	0

Table 2

As shown in Table 2, when the recovered data signal (RC_{in}) is less than all three threshold values, the symbol generated is 00 (MSB = 0, LSB = 0). Thereafter, as each of the three threshold values is exceeded, a different symbol is generated, as shown in the table above.

The MSB output from the 4-level decoder **810** is coupled to an input of the symbol synchronizer **812** and provides a recovered data input generated by detecting the zero crossings in the 4-level recovered data signal. The positive level of the recovered data input represents the two positive deviation excursions of the analog 4-level recovered data signal above the average threshold output signal, and the negative level represents the two negative deviation excursions of the analog 4-level recovered data signal below the average threshold output signal.

The symbol synchronizer **812** uses a 64X clock at 102.4 KHz which is generated by frequency divider **846**, that is coupled to an input of a 32X rate selector (not shown). The 32X rate selector is preferably a divider which provides selective division by 1 or 2 to generate a sample clock which is thirty-two times the symbol transmission rate. A control signal (1600/3200) is coupled to a second input of the 32X rate selector, and is used to select the sample clock rate for symbol transmission rates of 1600 and 3200 symbols per second. The selected sample clock is coupled to an input of 32X data oversampler (not shown) which samples the recovered data signal (MSB) at thirty-two samples per symbol. The symbol samples are coupled to an input of a data edge detector (not shown) which generates an output pulse when a symbol edge is detected. The sample clock is also coupled to an input of a divide-by-16/32 circuit (not shown) which is utilized to generate 1X and 2X symbol clocks synchronized to the recovered data signal. The divided-by-16/32 circuit is preferably an up/down counter. When the data edge detector detects a symbol edge, a pulse is generated which is gated by an AND gate with the current count of divide-by-16/32 circuit. Concurrently, a pulse is generated by the data edge detector which is also coupled to an input of the divide-by-16/32 circuit. When the pulse coupled to the input of the AND gate arrives before the generation of a count of thirty-two by the divide-by-16/32 circuit, the output generated by the AND gate causes the count of divide-by-16/32 circuit to be advanced by one count in response to the pulse which is coupled to the input of divide-by-16/32 circuit from the data edge detector, and when the pulse coupled to the input of the AND gate arrives after the generation of a count of thirty-two by the divide-by-16/32 circuit, the output generated by the AND gate causes the count of divide-by-16/32 circuit to be retarded by one count in response to the pulse which is coupled to the input of divide-by-16/32 circuit from the data edge detector, thereby enabling the synchronization of the 1X and 2X symbol clocks with the recovered data signal. The symbol clock rates generated are best understood from Table 3 below.

Input Clock (Relative)	Control Input (SPS)	Rate Selector Divide Ratio	Rate Selector Output	2X Symbol Clock (BPS)	1X Symbol Clock (BPS)
64X	1600	by 2	32X	3200	1600
64X	3200	by 1	64X	6400	3200

Table 3

As shown in the table above, the 1X and 2X symbol clocks are generated 1600, 3200 and 6400 bits per second and are synchronized with the recovered data signal.

The 4-level binary converter **814** couples the 1X symbol clock to a first clock input of a clock rate selector (not shown). A 2X symbol clock is coupled to a second clock input of the clock rate selector. The symbol output signals (MSB, LSB) are coupled to inputs of an input data selector (not shown). A selector signal (2L/4L) is coupled to a selector input of the clock rate selector and the selector input of the input data selector, and provides control of the conversion of the symbol output signals as either 2-level FSK data, or 4-level FSK data. When the 2-level FSK data conversion (2L) is selected, only the MSB output is selected which is coupled to the input of a conventional parallel to serial converter (not shown). The 1X clock input is selected by clock rate selector which results in a single bit binary data stream to be generated at the output of the parallel to serial converter. When the 4-level FSK data conversion (4L) is selected, both the LSB and MSB outputs are selected which are coupled to the inputs of the parallel to serial converter. The 2X clock input is selected by clock rate selector which results in a serial two bit binary data stream to be generated at 2X the symbol rate, which is provided at the output of the parallel to serial converter.

Referring again to FIG. 8, the serial binary data stream generated by the 4-level to binary converter **814** is coupled to inputs of a synchronization codeword correlator **818** and a demultiplexer **820**. Predetermined "A" codeword synchronization patterns are recovered by the controller **816** from a code memory **822** and are coupled to an "A" codeword correlator (not shown). When the synchronization pattern received matches one of the predetermined "A" codeword synchronization patterns within an acceptable margin of error, an "A" or "A-bar" output is generated and is coupled to controller **816**. The particular "A" or "A-bar" codeword synchronization pattern correlated provides frame synchronization to the start of the frame ID codeword, and also defines the data bit rate of the message to follow, as was previously described.

The serial binary data stream is also coupled to an input of the frame codeword decoder (not shown) which decodes the frame codeword and provides an indication of the frame number currently being received by the controller **816**. During sync acquisition, such as following initial receiver turn-on, power is supplied to the receiver portion by battery saver circuit **848**, shown in FIG. 8, which enabled the reception of the "A" synchronization codeword, as described above, and which continues to be supplied to enable processing of the remainder of the synchronization code. The controller **816** compares the frame number currently being received with a list of assigned

frame numbers stored in code memory **822**. Should the currently received frame number differ from an assigned frame numbers, the controller **816** generates a battery saving signal which is coupled to an input of battery saver circuit **848**, suspending the supply of power to the receiver portion. The supply of power will be suspended until the next frame assigned to the receiver, at which time a battery saver signal is generated by the controller **816** which is coupled to the battery saving circuit **848** to enable the supply of power to the receiver portion to enable reception of the assigned frame.

A predetermined "C" codeword synchronization pattern is recovered by the controller **816** from a code memory **822** and is coupled to a "C" codeword correlator (not shown). When the synchronization pattern received matches the predetermined "C" codeword synchronization pattern with an acceptable margin of error, a "C" or "C-bar" output is generated which is coupled to controller **816**. The particular "C" or "C-bar" synchronization codeword correlated provides "fine" frame synchronization to the start of the data portion of the frame.

The start of the actual data portion is established by the controller **816** generating a block start signal (Blk Start) which is coupled to inputs of a codeword de-interleaver **824** and a data recovery timing circuit **826**. A control signal (2L / 4L) is coupled to an input of clock rate selector (not shown) which selects either 1X or 2X symbol clock inputs. The selected symbol clock is coupled to the input of a phase generator (not shown) which is preferably a clocked ring counter which is clocked to generate four phase output signals (Ø1-Ø4). A block start signal is also coupled to an input of the phase generator, and is used to hold the ring counter in a predetermined phase until the actual decoding of the message information is to begin. When the block start signal releases the phase generator, it begins generating clocked phase signals which are synchronized with the incoming message symbols.

The clocked phase signal outputs are then coupled to inputs of a phase selector **828**. During operation, the controller **816** recovers from the code memory **822**, the transmission phase number to which the messaging unit is assigned. The phase number is transferred to the phase select output (Ø Select) of the controller **816** and is coupled to an input of phase selector **828**. A phase clock, corresponding to the transmission phase assigned, is provided at the output of the phase selector **828** and is coupled to clock inputs of the demultiplexer **820**, block de-interleaver **824**, and address and data decoders **830** and **832**, respectively. The demultiplexer **820** is used to select the binary bits associated with the assigned transmission phase which are then coupled to the input of block de-interleaver **824**, and clocked into the de-interleaver array on each corresponding phase clock. In a first embodiment, the de-interleaver uses an 8 x 32 bit array which de-interleaves eight 32 bit interleaved address, control or message codewords, corresponding to one transmitted information block. The de-interleaved address codewords are coupled to the input of address correlator **830**. The controller **816** recovers the address patterns assigned to the messaging unit, and couples the patterns to a second input of the address correlator. When any of the de-interleaved address codewords matches any of the address patterns assigned to the messaging unit within an acceptable margin of error (e.g., the number of bit errors correctable

according to the codeword structure selected), the message information and corresponding information associated with the address (e.g., the information representing the broadcast and received selective call signaling message, which was previously defined as message related information) is then decoded by the data decoder **832** and stored in a message memory **850**.

Following the detection of an address associated with the messaging unit, the message information is coupled to the input of data decoder **832** which decodes the encoded message information into preferably a BCD or ASCII format suitable for storage and subsequent display.

Alternatively, the software based signal processor may be replaced with a hardware equivalent signal processor that recovers the address patterns assigned to the messaging unit, and the message related information. Following, or prior to detection of an address associated with the messaging unit, the message information and corresponding information associated with the address may be stored directly in the message memory **850**. Operation in this manner allows later decoding of the actual message information, e.g., that encoded message information that decodes into a BCD, ASCII, or multimedia format suitable for subsequent presentation. However, in performing direct storage, the memory must be structured in a manner that allows efficient, high speed placement of the message information and corresponding information associated with the address. Additionally, to facilitate the direct storage of message information and corresponding information associated with the address in the message memory **850**, a codeword identifier **852** examines the received codeword to assign a type identifier to the codeword in response to the codeword belonging to one of a set comprising a vector field and a set comprising a message field. After determining the type identifier, a memory controller **854** operates to store the type identifier in a second memory region within the memory corresponding with the codeword. The above memory structure and operation of the de-interleaved information memory storage device comprising the message memory **850**, the codeword identifier **852**, and the memory controller **854**, are more fully discussed in the patents incorporated below.

Following the storage of the message related information, a sensible alert signal is generated by the controller **816**. The sensible alert signal is preferably an audible alert signal, although it will be appreciated that other sensible alert signals, such as tactile alert signals, and visual alert signals can be generated as well. The audible alert signal is coupled by the controller **816** to an alert driver **834** which is used to drive an audible alerting device, such as a speaker or a transducer **836**. The user can override the alert signal generation through the use of user input controls **838** in a manner well known in the art.

The stored message information can be recalled by the user using the user input controls **838** whereupon the controller **816** recovers the message information from memory, and provides the message information to a display driver **840** for presentation on a display **842**, such as an LCD display.

In addition to the preceding description, the systems previously discussed in reference to FIGs. 1, 2, 7 and 8, and protocol previously discussed in reference to FIGs. 3, 4 and 5 may be more fully understood in view of the following United States Patents: No. 5,168,493 entitled "Time

Division Multiplexed Selective Call System" issued to Nelson et al., No. 5,371,737 entitled "Selective Call Receiver For Receiving A Multiphase Multiplexed Signal" issued to Nelson et al., No. 5,128,665 entitled "Selective Call Signalling System" to DeLuca et al., and No. 5,325,088 entitled "Synchronous Selective Signalling System" to Willard et al., all of which are assigned to Motorola, Inc., and the teachings of which are incorporated herein by reference thereto.

Referring to FIG. 9, a diagram shows a selective call messaging system **900** capable of communicating compressed messages in accordance with the present invention.

The paging terminal **102** or wireless selective call signaling system controller, receives information comprising a selective call message request including original text and a destination identifier. The information is typically coupled to the paging terminal **102** via a Public Switched Telephone Network (PSTN) **912** which serves to transport the information from an originator **914**. The PSTN **912** may be coupled to the paging terminal **102** and the originator **914** using conventional phone lines **910** or possibly a high speed digital network, depending on the information bandwidth required for communicating messages between the originator **914** and a plurality of messaging units **906**. Once coupled to the paging terminal **102**, the information is compressed and formatted as one or more selective call messages that are transferred **922** to at least one radio frequency transmitter **904** for broadcast to at least one messaging unit **906** located in any one of a number of communication zones **902**. The messaging unit **906** operates to decompress the compressed message and store the decompressed message in message memory.

Two-way capability may be provided for the messaging unit **906** using either a wired or a wireless return path. By way of example, the message is received by the messaging unit **906** which decompresses a content of the message. This message content is then stored by the messaging unit **906** pending presentation. Using a reverse or inbound channel, the messaging system allows wireless return or origination of messages received by distributed receiver sites **908**. These sites are typically more dense that the outbound broadcast sites **904** since the transmitter power and antenna characteristics of the messaging unit **906** are typically inferior to that of a dedicated radio frequency base station and wide area transmitter site **904**. Thus, the size and weight of each messaging unit **906** is kept to a minimum, yielding a more ergonomic portable device with the value added function of not requiring a physical connection to a wired network when effecting messaging. Alternatively, the secure messaging system is adapted to accommodate lower power messaging unit **906** devices that might include additional means for implementing the return or origination of messages using a reverse or inbound channel that is accessed at specific points in the network. In these cases, the lower power messaging unit **906** could include an infrared or laser optical port, low power proximate magnetic inductive or electric capacitive port, or possibly an ultrasonic or audio band acoustic transducer port, all of which could couple signals between the lower power messaging unit **906** and a distributed communication device or the like. In certain cases, this short-range localized type of transmission might be desirable because of privacy or security concerns.

In view of the preceding discussion, the remainder of the compressed messaging system is described with reference to FIGs. 10-16.

Referring to FIG. 10, the illustration shows a high level block diagram of a messaging unit **906** in accordance with the preferred embodiment of the present invention.

The preferred embodiment of the messaging unit **906** is a conventional paging device as shown in FIG. 10 modified to include a message decompression engine **1014** and associated memory **1016** containing compression token table(s) and other data elements necessary for message decompression. The electronics required to implement the decompression engine **1014** may be integrated with the paging device. Alternatively, the message decompression engine may be implemented as an application in software or firmware that is executed using the central processing unit (CPU) **1006**, random access memory (RAM) **1008** and read-only memory (ROM) **1010**.

When operating in the receive mode, the incoming signal is captured by the antenna **802** coupled to the receiver **804** which detects and demodulates the signal, recovering any information as previously discussed with reference to FIG. 8.

Alternatively, when operating as a messaging unit with origination capability, the messaging unit **906** may contain a low power reverse channel transmitter **1034**, power switch **1032**, and transmit antenna **1030** for either responding to an outbound channel query or generating an inbound channel request. Instead of the portable transmitter **1034** (e.g., a low power radio frequency device) and its associated components, the alternative transmission block **1036** may additionally contain either uni- or bi-directional communication transducers. Examples of such transducers are optical devices like lasers or light emitting diodes (LED), extremely low power magnetic field inductive or electric field capacitive structures (e.g., coils, transmission lines), or possibly acoustic transducers in the audio or ultrasonic range.

An optional input/output (I/O) switch **1002** serves to direct the incoming or outgoing radio frequency (RF) energy between the RF receiver **804**, RF transmitter **1034** and a selective call decoder **1004**. The selective call decoder **1004** comprises the CPU **1006**, and its associated RAM **1008**, ROM **1010**, and universal input/output (I/O) module **1012**. The primary function of the selective call decoder **1004** is to detect and decode information contained in signaling intended for receipt by the messaging unit **906**. However, in a two-way implementation that includes the optional reverse channel transmitter block **1036**, the selective call decoder **1004** can function as an encoder to generate and deliver requests or messages to the originator **914**, a user, or other on-line system (not shown). Additionally, as previously mentioned, the components within the selective call decoder **1004** may be used to implement the message compression/decompression engine.

In the case where the messaging unit **906** acts like a remote paging terminal with origination capability, the messaging unit **906** can further operate as a message generator to create a outgoing selective call message and its corresponding system transmission request. An optional message entry device **1018** allows a user to initiate a selective call message or the like. Typically, a user might enter a request using a keyboard, a voice activated recognition device, a touch-

sensitive device (e.g., screen or pad), or other convenient data entry device. Once created, a portable transmitter **1034** coupled to the message generator operates to broadcast the selective call message request for receipt by one of the distributed receiver sites **908** and coupling to a paging terminal for decoding, forwarding, or any other function requested by the originator. Similarly, a user may return a message in response to prior messages stored in the messaging unit **906** or information recently communicated with the messaging unit **906**. In this way, the messaging unit **906** can initiate messaging transactions without requiring a physical connection to a land-line hard wired network or PSTN.

With regard to the implementation of a radio frequency enabled reverse channel messaging unit **906** as discussed herein, the invention preferably operates using the Motorola ReFlex™ two-way wireless paging system infrastructure and protocol which is described in detail in the following documents: U.S. patent application number 08/131,243, filed October 4, 1993 by Simpson et al. and titled "Method And Apparatus for Identifying a Transmitter in a Radio Communication System"; U.S. patent application number 08/398,274, filed March 3, 1995 by Ayerst et al. and titled "Method And Apparatus for Optimizing Receiver Synchronization in a Radio Communication System"; U.S. patent number 5,521,926 issued May 28, 1996 to Ayerst et al. and titled "Method And Apparatus for Improved Message Reception at a Fixed System Receiver"; U.S. patent application number 08/498,212, filed July 5, 1995 by Ayerst et al. and titled "Forward Channel Protocol to Enable Reverse Channel Aloha Transmissions"; and U.S. patent application number 08/502,399, filed July 14, 1995 by Wang et al. and titled "A System and Method for Allocating Frequency Channels in a Two-way Messaging Network", all of which are assigned to the assignee of the present invention, and all of which are incorporated by reference herein.

It should be appreciated that the use of the instant invention in other two-way communication systems such as cellular and radio packet data systems is contemplated.

Referring to FIG. 11, the block diagram illustrates message composition and compression equipment that could be used to send compressed messages to subscriber messaging units via a paging channel or the like.

Specifically, both direct branch and customer calls are received by a selective call processor **1100** comprising a message processing computer **1102**, a message compression computer **1104**, a subscriber database **1106**, and a compression token database **1108**. The message processing computer **1102** receives messaging requests and communicates with the message compression computer **1104** to generate and compress the original text (numeric or alphanumeric) message based on information (e.g., tokens) contained in the compression token database **1108**. The message processing computer **1102** also determines a destination identifier from information contained in the subscriber database **1106**, which allows a selective call message distributor (an intrinsic function of the selective call processor) to communicate the destination identifier, typically as a selective call address, and its corresponding compressed message, to a selective call transmission service **904**. The destination identifier may correspond with a conventional paging or

selective call address, a cellular telephone address, or any other address that uniquely identifies a destination associated with the compressed message. Additionally, the subscriber database **1106** may include a compression version identifier that corresponds with a specific destination identifier. The purpose of the compression version identifier is to allow each messaging unit to operate with either the same compression and decompression procedure, or with compression and decompression procedures that are specifically adapted to the text, e.g., characters or language, being communicated between messaging units or alternate devices of like capability. As an example, if the messaging unit is primarily used with English text, then the compression version identifier would indicate the version of the compression and decompression procedure used to compress and/or decompress the English text communicated with the messaging unit. Similarly, if the messaging unit is primarily used with Mexican Spanish text, as opposed to Spain's Spanish text, then the compression version identifier would indicate the version of the compression and decompression procedure used to compress and/or decompress the Mexican Spanish text communicated with the messaging unit.

The message composition and compression equipment illustrated in FIGs. 11 and 12 would typically be used on the premises of a radio common carrier or other messaging service provider to send compressed messages to messaging units **906** via a conventional paging channel or the like. However, many alternate implementations are possible in which the message entry, receipt, capture, generation, and compression may be distributed in any number of ways. For example, a software application program executing on a conventional personal computer (PC) may implement the data entry function using an associated keyboard or other data entry device. The same PC may have access to a local or remote subscriber database with selective call address information. Finally, the software application program may include the procedures and associated token tables needed to compress the text associated with the selective call message. In this example, the message or messages delivered to a paging terminal may already be pre-compressed and addresses, so that the paging terminal need only forward the compressed messages to the particular messaging units.

Referring to FIG. 12, the illustration shows a functional diagram of a wireless selective call signaling system controller that implements a combined one-way and two-way compressed messaging system capable of signaling the messaging units.

The wireless selective call signaling system controller comprises the paging terminal **102** along with a transmitter **104** and associated antenna **904**, and in two-way RF systems, at least one receiver **1202** system comprising a received signal processor **1204** and at least one receive antenna **908**. Preferably, several of at least one receiver **1202** systems may be distributed over a wide geographical area to receive the low power transmissions broadcast by two-way messaging units **906**. The number of receiver **1202** systems in any given geographical area is selected to insure adequate coverage for all inbound transmissions. As one of ordinary skill in the art will appreciate, this number may vary greatly depending on terrain, buildings, foliage, and other environmental factors.

The wireless selective call signaling system controller represents a closely coupled implementation of the overall compressed messaging system. In practice, an originator is not the party responsible for maintaining the RF infrastructure, i.e., the transmitter **104** and associated antenna **904**, and the at least one receiver **1202** system. Consequently, a conventional wireless messaging service provider or radio common carrier would provide and maintain the RF infrastructure, and the originator would utilize that RF infrastructure in a conventional manner to communicate compressed messages between the originator and the messaging units **906**.

As a first alternative to the preceding operation, the selective call signaling system controller may operate to compress, encode, and transmit compressed messages received from an originator, where the selective call processor **1100** has generated the compressed message. Subsequently, the messaging unit **906** decodes and decompressed the compressed message, revealing the original text message. Similarly, the selective call signaling system controller may receive messages originating from the messaging unit **906**, decode and decompress the message, then pass the decompressed message to the originator as a reply to an original message sent to the messaging unit **906**.

As a second alternative to the preceding operation, the selective call signaling system controller may operate to encode and transmit compressed messages communicated between the originator and the messaging unit **906**. In this case, the originator may pre-compress the original text message and forward it along with a destination identifier to the selective call processor **1100**. The selective call signaling system controller then operates to associate a selective call address with the pre-compressed message based on a received destination identifier, and transmit a resulting compressed selective call message for receipt by the messaging unit **906**. Subsequently, the messaging unit **906** decodes and decompresses the selective call message, revealing the original text message. As with the prior operation, the selective call signaling system controller may operate to receive messages originating from the messaging unit **906** and pass the received message in its compressed or decompressed state to the originator as a reply to an original message sent to the messaging unit **906**.

Additionally, in each of the preceding cases, the messaging unit **906** may operate to originate a compressed or uncompressed message targeted for either another messaging unit or any device capable of processing the message for presentation.

The preferred embodiment of the present invention implements a text compression procedure that is described in detail in the following text. Since most messages in conventional selective call messaging systems are relatively short e.g., from tens of characters to several hundred characters, a static token table design has been selected that gives good efficiency in both coding size and execution (compression / decompression) speed. Several test cases were performed on statistical data sets using the instant procedure, yielding compression on English and Spanish test data sets of 1.534 and 1.878 respectively. These compression rates correspond with throughput increases of at least fifty percent for English text messaging and at least eighty percent for Spanish text messaging.

To maintain flexibility in the compression system, the defined data types are re-definable, as is the data switch. To further insure compatibility, an application version number may be specified in the message control segment of a compressed message.

No matter how big a token table is, there is always be a large portion of messages that cannot be encoded using the tokens available in the selected table. Consequently, an efficient representation of an original text message, e.g., character string, is crucial to achieving a good compression ratio with this procedure. For purposes of illustration, the character string is defined as follows:

<Character String> ::= a | b | ...z | 0 | 1 | .. | 9 | <space> | <.> | <,> | </> | ...

where:

Length: 2 or 3
Coding: 6 bits fixed
Table size: 64 elements
Leading space: none

According to this definition, there are 64 elements (characters) using 6 bits each in the hypothetical token table. The field size is fixed at 2-bits in length. Accordingly, 1 to 4 <Character String> (<cs>) length elements can be specified. If the length is longer than or equal to 4, an extra continuation bit needs to be added. The value '1' of the continuation bit indicates that there are more than 4 elements and the value '0' means there are 4 or less elements. In the case of bit '1', the decompressor is expecting another length field after the current four <cs> elements have been decoded. In theory, this process can go on forever.

The elements in the hypothetical <cs> character string token table for both English and Spanish are as follows:

Token	ID	Token	ID	Token	ID	Token	ID
<sp>	0	1	16	[	32	m	48
!	1	2	17	]	33	n	49
"	2	3	18	^	34	o	50
#	3	4	19	_	35	p	51
%	4	5	20	a	36	q	52
&	5	6	21	b	37	r	53
'	6	7	22	c	38	s	54
(	7	8	23	d	39	t	55
)	8	9	24	e	40	u	56
*	9	:	25	f	41	v	57
+	10	;	26	g	42	w	58
,	11	<	27	h	43	x	59
-	12	=	28	I	44	y	60
.	13	>	29	j	45	z	61
/	14	?	30	k	46	{	62
0	15	@	31	l	47	}	63

Table 4

In order to effectively code numeric information, an alternate token must be selected. In the present invention, a 4-bit mini token has been selected because all the digits and several other symbols associated with telephone numbers within an alphanumeric message are efficiently represented by 4-bit mini tokens. For purposes of illustration, the 4-bit mini token is defined as follows:

<mini-4> :: = 0 | 1 | ..| 9 | <sp> | <,> | <-> | <.> | </> | <:>

where:

Length type: 4 or 5
Coding: 4-bits fixed
Table size: 16
Leading space: none

There are preferably sixteen elements in the <mini-4> token table. Thus, a 4-bit length field is defined for the <mini-4> covering the length of one to sixteen. In case where sixteen is specified in the length field, another bit is added to specify the continuity of the mini-tokens. For example, for a <mini-4> string of length 16, the length field is '1111' with a continuity bit of '0'. If the length of the <mini-4> string is longer than 16, the continuity bit is set to '1' and another length field (4 or 5 bits) is expected after the 16 <mini-4> tokens are encoded.

The preferred <mini-4> token table is shown in Table 10.

In a manner similar to that described with respect to the <mini-4> token, a <mini-5> token is defined as follows:

<mini-5> :: = returned your call | please call | go direct | ext | please | pls. | pls | ...

Length: 1 or 2
Coding: 6 bits fixed
Table size: 32
Leading space: yes

There are preferably thirty two elements in the <mini-5> token table. Up to two bits are used for specifying the length of this data type. One bit is used for the length field. A '0' means length of one and '1' means length of two. If the length field is '0' no second bit is needed. If the bit is '1', a continuation bit is required ('0' means finish and '1' means continue). Two bits are sufficient because most of the consecutive <mini-5>s are short. The most significant bit (MSB) of <mini-5> is the leading space indicator, therefore, six bits are used for <mini-5> tokens as indicated in the preceding definition.

The preferred <mini-5> token table is shown in Table 11.

Table 5 shows a hypothetical <mini-5> token table for English as follows:

Token	ID	Token	ID
returned your call	0	dir	16
please call	1	ext	17
go direct	2	for	18
pls. call	3	pls	19
pls call	4	re:	20
tomorrow	5	am	21
at home	6	at	22
office	7	pm	23
please	8	re	24
pls cl	9	#	25
thanks	10	$	26
after	11	(	27
today	12	)	28
asap	13	a	29
call	14	f	30
pls.	15	p	31

Table 5

As discussed before, the compression procedure is designed to support multiple versions of compression and decompression. For example, two types of Spanish language messaging units may be defined. Type A messaging units would display English characters while Type B messaging units would display Spanish characters. This variation is necessary because paging terminals typically use different character sets as in the United States and Spain.

All of the Spanish tables shown here in this documentation can be used for both Type A and Type B messaging units. The special Spanish character strings (tokens) for Type B messaging units are listed in the following Spanish token tables in the parentheses. For example, token "manana a las" in the <Mini-5> token table for Type A Spanish messaging unit is replaced by "*mañana* a las" for the Type B Spanish messaging unit.

Token	ID	Token	ID
este radio. comunicarse	0	con la	16
favor comunicarse al	1	de las	17
favor de comunicarse	2	espero	18
comunicarse al	3	a las	19
comunicate al	4	para	20
antes de las	5	com	21
manana(mañana) a las	6	con	22
comunicarse	7	ext	23
urgente al	8	tel	24
hasta las	9	].	25
urgente.	10	al	26
com. al	11	de	27
urgente	12	*	28
com al	13	[	29
com.al	14	]	30
con el	15	y	31

Table 6

A token table is implemented that includes tokens comprising all ASCII characters except upper case and the most frequently used phrases, words and fragments. The selected token table size is 542 elements with each element having a fixed length of 10 bits. The most significant bit (except for command tokens) is a defined as a leading space indicator. The following relation describes the general structure and contents of the token table.

Token table :: = command tokens | ASCII | phrases | fragments

where:

Length: arbitrary (0)
Coding: 10 bit fixed
Table size: 542
Leading space: partial

and the <Token Table> (<T>) is the default data type.

In the present implementation, there are 60 command tokens consisting of capital control tokens and data type switch tokens. This number may be increased of decreased depending on the size of the data set being considered and the compression desired. In the preferred embodiment, address fields 0 to 29 and 512 to 541 are reserved for command tokens. One of ordinary skill in the art will readily realize that the selected fields and reserved token choices may be modified without deviating from the spirit of the present invention, so long as the general rules disclosed in this procedure are followed.

Capital control tokens are used to control the capitalization of characters, words, or phrases. In the preferred embodiment of the present invention, there are three possible capital control tokens: <all cap>, <all low>, <cap first>. The capital control tokens are denoted as data type <cc>.

<all cap> indicates that all following alphabetical characters in a string are to be capitalized. This token has no effect on characters such as numerals, '0' to '9', and symbols, ',', '$', etc. <all cap> is the default capitalization token, i.e., if no capital control token shown in the encoded message, the decoded message will be all capitalized or the decoded message will be capitalized until one of other capital control tokens is encountered.

<all low> indicates that the following alphabetical characters are in lower case.

<cap first> indicates that the first alphabetical character encountered is capitalized and the following characters in lower case.

If a capital control token is used adjacent to any data type other than <T>, it is represented as data type '0' during the shortest path encoding. For example the data type switch '40c' represents a data type switch of <mini-4> <capital control> (<cc>) and <character string> (<cs>). The hypothetical coding of <cc> is defined as:

<cc> :: = <all cap> | <all low> | <cap first> | (<reserved>)

where:

Length: 0
Coding: 2 bits fixed
Table size: 3 with 1 reserved
Leading space: none

There are three (plus one reserved) elements in the data type. The length is fixed to one. Two bits are used to select one of the three or possibly four choices. Otherwise, if a capital control token is used apart from a <mini-4> or <mini-5> or <cs>, a distinct token is used for each of the individual capital control tokens. For example, <all cap> is '6', <all low> is '7' and <cap first> is '8'. Note that if a message is all capitals, the text compression rate is further improved.

The preferred capitalization control token table is shown in Table 13.

A set of data type switch control tokens are defined in the token table. Since the token table serves as the default data type in the compression process, only the data type switch among <mini-4>, <mini-5>, <capital control> and <character string> are included in the data type switch control token set. The selection of the data type switch tokens is based on the merit of the tokens and is defined as follows.

<command tokens> ::= cap control tokens | data type switch tokens

where:

Length field: 0
coding: 10 bit fixed
Table size: 60
Leading space: none

The preferred command tokens are listed in the command token Table 13, Table 20 and Table 21.

The phrase and fragment token table contains a number of language phrases and fragments depending on the statistics of the information being compressed. In the present embodiment, the phrase and fragment token table comprises 406 phrase and fragment tokens and 76 ASCII tokens in the token table. By definition, the phrase and fragment token table is as follows.

<phrase and fragment> ::= 406 phrase and fragment tokens | 76 ASCII characters

where:

Length: 0
Coding: 10 bits fixed
Table size: 482 elements
Leading space: yes

No distinction is made between a phrase and fragment either during token selection or during the shortest path encoding of a message. The token selection process selects the right number of phrase and fragment tokens automatically. In this way, an orthogonal property among tokens is guaranteed which leads to the highest compression rate possible.

After the data types are defined, an efficient compression (encoding) process is accomplished as follows. First, long messages are fragmented to improve the performance of the encoding process. According to the encoding process, a fragment is generated before a <mini-4> token or when a <capital control> token is encountered. <cc> and <mini-4> tokens are selected to fragment a message because (1) they occur frequently, (2) there are a very limited number of data types to represent the elements in these tables, and (3) generally, they are the most efficient representation of the information. This inherently yields an efficient code which results in optimal compression of the text data. For example, there is only one way to represent all the elements in <cc> table (that is <cc>), but there are up to three ways to represent an element in the <mini-4> table (<mini-4>, <cs> and <T>). Alternatively, a fragment can be generated before a token of a special character not covered by any of the <character string>, <mini-5> or <mini-4> (e.g. '!') or before any token in the token table if none of above mentioned criteria is satisfied.

Second, the <mini-4> must be identified. For example, suppose the length of a possible <mini-4> string is L. The savings of using <mini-4> over using 10 bit tokens can be expressed as $S = 10 \times L - (10 + 4 + 4L) = 6L - 14$. In other words, a <mini-4> of a length greater than two will save some bits over <T> in any case. Thus, to effectively fragment a message, the process identifies all <cc> tokens and the <mini-4> tokens with a length greater than two, then the message is fragmented accordingly.

Third, to maximize coding efficiency, a shortest path procedure was designed for the encoding process. The major difference between the instant shortest path procedure and a conventional shortest path procedure is that the present procedure is based on a context-dependent cost matrix while the prior art is based on a context-independent cost matrix.

In the proposed shortest path procedure, the cost (number of bits used) is dependent on the history of data type switches and the length of each of the individual data type.

For example, if the current pending encoding data type is <mini-4> and the previous data type is <T>, ten bits are needed for the data type switch. On the other hand, if this is the second data type in the data type switching chain beginning from a <T> at the second previous node and its previous data type is <cs>, and there is a data type switch token in the command token table with value 'c4', the incremental cost for the current data type switch is zero (no extra bit is needed for introducing another data type switch <mini-4> after <cs> at this particular node).

In the preferred embodiment, the cost within a particular data type is also a function of the length of the data type. For example, if the current data is <mini-4> and it is the third consecutive token of the same data type, the incremental cost of encoding this data is 4 bits. On the other hand, if every element is the same except that this is the 16th consecutive <mini-4> data token, the cost will be 5 instead of 4. The extra bit is for the continuation bit in the length field.

Since <T> serves as the default data type, there are four different data types involved in the data type switch tokens. They are <mini-4>, <mini-5>, <cs> (character string) and <cc> (capital control) tokens.

For illustration purposes, a data type switch length parameter L is defined. L is assigned a value of '0' when it is data type <T>. Whenever a data type is switched out from <T>, L is assigned a value of '1'. So long as the data type is not changed, the value of L is not changed. If a new data type switch is added (other than switching back to <T>, L is reset to '0') the value of L of the current node will be incremented.

The parameter L is the L value of the current node. The incremental cost function $C_i(L)$ is defined based on a simulated result as:

$$C_i(L) = \begin{cases} 0.00 & L < 4 \\ 3.10 & L = 4 \\ 1.33 & 4 < L < 8 \\ 2.59 & L = 8 \\ 2.06 & L > 8 \end{cases}$$

The cost $C_i(L)$ of zero for $L<4$ is justified considering the fact that ten bits are needed for any initial data type switch out of <T> and that there is a matching data type switch token for any data type switch with sequences of length one to three with very high probability (which is ensured by the data type switch token selection procedure).

The above defined $C_i(L)$ is an estimation of the incremental data type switching cost. Based on $C_i(L)$, the accumulative data type switching cost $C_a(L)$ is estimated. For example, $C_a(5)=10+3.1+1.33 = 14.43$. This example indicates that the estimated cost for a data type switch with length 5 is 14.43 bits.

To ensure a good compression rate, a better estimation of the data type switching cost must be made in addition to $C_a(L)$. For example, consider the matching between the data type switch sequence and the data type switch tokens. A data type sequence (S = DTSS) is subsequently defined for all non-<T> data type nodes in the shortest path procedure. The S at a

node with data type <T> is then defined as empty. The S should be updated whenever a new data type switch occurs. If the ending data type is any data type other than <T>, the S of the ending data type node should be concatenated to the S of its previous node with the ending data type. For example, in the data type switching sequence, if there is <capital control> after <mini-5>, the <capital control> ('0' for short) code is then attached after the <mini-5> code ('5' for short) in the associated S of the current node, i.e., '50'.

C_t (S) is defined as a measurement of the cost (number of bits) of encoding data type switching sequence S. The value of the C_t (S) is calculated using the shortest path encoding procedure. For example, if S = '45054' and there happens to be a data type switch token '45054', then C_t (S) is 10 + 2 = 12. The first ten bits are for the data type switch token '45054' and the other two bits are for the <capital control> selection indicated by the data type '0'.

Now consider the fact that a partial string may cost more than a full string. For example, what if S = '45054c', and there are data type switch tokens in the token table '450', '54' and '45054c'. The partial string '45054' costs 20 bits while the full string '45054c' cost 10 bits. To improve the compression rate, an 'S prefix' concept is introduced.

The 'S prefix' sequence satisfies the following three conditions:

(a): it is the postfix sub-sequence of the associated S;
(b): it is the prefix sub-sequence of some of the selected data type switch tokens
(c): the length of the sequence should be greater than one.

For example, if S: '45c54c4', and a data type switch token is '4c454', the string '4c4' is the prefix sequence satisfying all the conditions mentioned above. If no such sequence exists for a given S, the prefix string is defined as empty with length zero.

Now, parameter L_{pre} is defined as the maximum length of the prefixes associated with a given S of length L. Accordingly, the estimated switching cost of S with length L is defined as:

$$C(S) = C_t(S) - H \times [C_t(S) - C_a(L - L_{pre}) - 10]$$

Where 'H' is a parameter to be optimized. It is defined that if $L_{pre} = 0$ or $C_a(L - L_{pre}) + 10 > C_t(S)$, $C(S) = C_t(S)$. In fact, C(S) is defined as a value in between $C_a(L - L_{pre}) + 10$ and $C_t(S)$ if the former is smaller than the latter. Otherwise, C(S) has the value of the latter.

In this case, the optimization process selected H = 0.65.

It should be noticed that the value of C(S) is cumulative and is based on the statistics of the data type switching cost of the system. C(S) will have a low cost if it could be encoded by small number of data type switch tokens or it has a long prefix sequence. Therefore C(S) is a desirable data type switching cost estimator.

In the forward shortest path encoding procedure the parameters L, the string S and C(S) are updated for all the non-<T> stages in all nodes. This in turn contributes to the cost calculation of each node. In a backward tracing process of the shortest path encoding procedure, the stage (data type) with the minimum cost value is selected.

The cost of switch from <T> to <mini-4> is C_{t4} = 10+4+4. The first 10 is the ten bit data type switching from <T> and the next four bits are the length field of <mini-4>. The last four bits are the <mini-4> token itself. The process sets n_4 = 1 (n_4 is the length of the <mini-4>) and S = '4'. The length for the <mini-4> sequence may be longer than 15. Thus more than 4 bits may be needed for the length fields. The additional bits for the length fields are considered in the cost of C_{44}.

In a similar way the other data type switching cost can be calculated.

C_{t5} = 10 + 1 + 6 = 17. The first ten bits are the data type switch. The next bit is for the length field ('0'). The following six bits are for the first token. Set S_5 = '5'. Since the incremental cost of <mini-5> is a constant 6, there is no need to keep track of the parameter n_5.

C_{tc} = 10 + 2 + 6 = 18. The first ten bits are for the data type switching. The next two bits are for the length field. The last six bits are for the character itself. Set n_c = 1, S_c = 'c'. To encourage the usage of <character string>, the overhead (2 bits for the length field) of the switching from <T> to <character string> is distributed in several characters, say two characters in this case. By doing so, C_{tc} = 10 + 1 + 6 = 17, C_{cc} = 6 + 1 = 7 for the second character and C_{cc} = 6 for the third of the <cs> and so on. Special attention is paid when there is only one character in the <character string> and when the data type switches to another data type.

$$C_{tt} = 10;$$

$$C_{44} = \begin{cases} 4 & o.w. \\ 5 & n = 15 \\ 8 & n = 0 \end{cases} \qquad n = (n+1) \bmod 16$$

It is clear that C_{44} is a function of n which is the sequence number of the current <mini-4> token in the <mini-4> sequence. The cost of C_{44} calculated first and then the value of n is updated.

C_{45}= 1 + 6 + C(S_4 + '5'); one bit is used for the length field for <mini-5> and six bits are used for the <mini-5> token. The data type switching cost C(S) is also considered. Due to the fact that the C_{45} is incremental in nature while the C(S) is accumulative, special attention is paid not to duplicate the C(S) in the cost estimation. Set $S_5 = S_4$ + '5'. Note, the S_4 is from its previous node.

Now define C_{4t} = 10; C_{4c}= 2 + 6 + C(S_4+ 'c') and set n_c = 1 and $S_c = S_4$ + 'c'.

Define C_{54} = 8 + C(S_5+ '4'), set n_4 = 1 and $S_4 = S_5$ + '4'.

C_{5t} = 10. C_{5c} = 8 + C(S_5 + 'c'), set n_c = 1 and $S_c = S_5$ + 'c'.

$$C_{55} = 7$$

In the same way, define C_{c4} = 8 + C(S_c + '4'), n_4 = 1, $S_4 = S_c$ + '4'; C_{c5} = 7 + C(S_c + '5'), and update $S_5 = S_c$ + '5'; C_{ct} = 10.

The cost of C_{cc} is then defined as:

$$C_{cc} = \begin{cases} 8 & l = 0 \\ 7 & l = 3 \\ 6 & o.w. \end{cases} \qquad l = (l+1) \bmod 4$$

Where l (equivalent to n_c) is the length of the character string before the current updating. Initially it is zero. Whenever a switch is made to <cs> from any of the <mini-4>, <cc>, <mini-5> or <T>, l is assigned a value of one. Any time a new character is added into the character string l is incremented by one modulo 4.

A summary of the cost of the data type switching is given in Table 7 below.

From \ To	<mini-4>	<mini-5>	<T>	<c string>
<mini-4>	C_{44}	7+C(S*)	10	8+C(S)
<mini-5>	8+C(S)	C_{55}	10	8+C(S)
<T>	10+8	10+7	10	18
<c string>	8+C(S)	7+C(S)	10	C_{cc}

Table 7

*: The data type switch sequence S in the table should be updated first.

Referring to FIG. 13, an exemplary state diagram shows the stages for a string of length n. For a length n string there are n + 2 stages in its state diagram. There is an initial and a terminal stage in the diagram. There is a link connecting two different stages in the state diagram if and only if there is a token in any of the above defined token tables which is identical to the that denoted by these two stages in the diagram. For example, the link connecting stage '0' and '11' in FIG. 13 is an indication that there is a token of "please call". It should be noticed that there may be more than one links between any two stages in the diagram. This happens in cases where there are a set of tables who share a common token. For example, each character in the message may have two ways to represent it. One is <cs>, another is <T>. This corresponds to having two links between any adjacent stage in the diagram. Note that only one link is shown in FIG. 13.

Referring to FIG. 14, the illustration shows a state table depicting a typical parsing and compression operation of the compression engine to resolve the shortest path for optimal text compression in accordance with the preferred embodiment of the present invention.

The shortest path is identified with the help of the shortest path spread sheet. An example of the spread sheet associated with the state diagram of FIG. 13 is shown in FIG. 14.

The first column of the spread sheet is the stage number. Each printable character has an unique stage number. The second column is the previous stage number of a associated link. There is a link for each character string which could be represented by a token in any of the token tables. There is a node for each of the links in the stage diagram. The third column is the character string represented by the link specific by the first two columns. The next three variables S_4, S_5, and S_c are the data type switch sequences associated with <mini-4>, <mini-5> and <cs> of a node

respectively. The data type switch sequence S has been defined before. The n_4 is the parameter n in the definition of C_{44} and the n_c is the parameter l of the definition of C_{cc}. The $\Sigma 4$ is the node minimum cost when the associated character string is treated as a <mini-4>. The node minimum cost is calculated by considering all the possible data types of its previous stage. Mathematically it is calculated as follows:

$$\sum 4 = \min(C_{44} + P_4, C_{54} + P_5, C_{t4} + P_t, C_{c4} + P_c)$$

where the P_k is the stage minimum cost of its previous stage in data type k which is one of the last four fields for that node. An infinity is assumed if a node has a previous stage of data type k with P_k = '-'. The C_{ij} s are defined before. The following three fields are calculated in a similar way.

Here is a good place to clarify some of the details of the calculation of the data type switching parameters S and C(S). There are three set of parameters for each node one for each of the data type other than the <T>. Namely, S_4 for <mini-4>, S_5 for <mini-5> and S_c for <character string>. These parameters should be updated according to the minimum cost calculation. For example, if the $\Sigma 4$ of the current node is derived from stage <mini-5> of its previous stage (i.e. $\Sigma 4 = C_{54} + P_5$), the S_4 of the current node should be S_5 + '4' where the S_5 is from its previous stage. If a node can not be treated as a <mini-4>, the value of S_4 is empty. This rule applies to all the other data type switching parameter set updating. For example, $\Sigma T = 10$ for node three stage one. It is derived as:

$$\sum T = \min(C_{4t} + P_4, C_{5t} + P_5, C_{tt} + P_t, C_{ct} + P_c) = \min(\infty, \infty, 10 + 0, \infty) = 10.$$

The previous stage of the node is stage zero.

If there are multiple links ended at a stage, there are multiple nodes under the stage. For example, there are two links ended at stage two therefore there are two nodes under stage two, each for a link. Define m as the link index of a stage. The stage minimum cost $S_i = (4, 5, T, c)$ is defined as the minimum of all the associated node minimum costs of that stage. This is mathematically represented as follows.

$$S_i = \min_m(\sum S_{i,m})$$

For example, the stage minimum cost T = 10 for the stage 6 in FIG. 14, that is the minimum of the three ΣT's with value 40, 30 and 10. It is noted that the field n_c for the character links are updated in the spread sheet. The trigger of updating is c = Σc for the stage and the rule of updating is defined in the following equation.

$$n_c = \begin{cases} 1 & \text{if switched from other data types} \\ n_c + 1 \bmod 4 & \text{otherwise} \end{cases}$$

For example, $n_c = 2$ for link one stage two of FIG. 14 since c = 24 = Σc and the Σc is derived from previous stage c to current stage c (17 + 7 = 24) and according to the above equation we have: $n_c = 2 = (1 + 1)$ modulo 4.

For each segment of a message the segment ending data type is selected at the end of the segment. For example, the segment ending data type is selected as a <mini-5> in the example shown in FIG. 14.

The selection of the segment ending data type depending not only on cost evaluation within its segment but also on the initial stage data type of its following segment. The data type of the ending stage of a segment is actually the data type of the initial stage of the following segment. Data type <T> is assumed in the example, as the initial stage data type of the following segment. Therefore, <T> is assigned as the unique data type of the ending stage 12. The <mini-5> is selected as the segment ending stage data type only when the following relation is evaluated:

$$\sum T = \min(C_{4t} + P_4, C_{5t} + P_5, C_{tt} + P_t, C_{ct} + P_c) = \min(\infty, 17+10, 20+10, 44+10) = 27.$$

The encoding can be easily identified in this example to be a <mini-5> token "please call".

The segment ending data type selection will follow the same rule as that specified in this section for other ending stage data types. The ending stage data type assignment becomes a don't care if the associated fragment is the last one of a message.

Referring to FIG. 15, an exemplary state diagram shows a typical initial data type assignment in accordance with the preferred embodiment of the present invention.

After a fragment is encoded the segment ending data type of the fragment serves as the initial stage data type of next fragment. In case a fragment is the first fragment of a message the default data type (e.g. <T>) serves as the initial stage data type as shown in FIG. 14.

Suppose, the following segment after "Please call" is "<>pm". Thus the initial stage of the associated state diagram of the second segment should be as shown in FIG. 15.

Due to the fact that the <T> is the default data type, the data type switch patterns among <mini-4>, <mini-5>, <cs> are collected as ten bit tokens to improve the compression rate. For example ten bits are used for a data type switch token '4545c'. That is equivalent to a sequence of

five data type switch tokens, namely, '4', '5', '4', '5' and 'c'. Thus forty bits are saved by using the data type switch token in the example.

In the encoding process a string of <mini-4> is treated as a single <mini-4> as far as the data type is concerned. The same treatment is performed in the encoding process for <mini-5> and <cs> strings. It is irrelevant to have a string of <cc>s.

The selection of data type switch tokens are based on the merit of each tokens. A priority function value is defined for each of the data type switch tokens: PFV = F * (L-1), where the F is the frequency of usage of the token in the encoding process and the L is the length of the pattern. For example, <mini-4><mini-5><cs> is of length 3. The top M data type switch tokens with the highest PFVs are selected for the data type switch token table, where the M is the size of the table.

To improve the error recovery capability of the system a bit padding technique is implemented. The technique requires bit padding in returning from the non-ten tokens to <T> so that the system can be recovered from any detectable or un-detectable errors as long as all the <T> tokens are beginning on the boundary of multiple of 10 bits. It is also noted that the error recovery is most desirable for long messages. The cost introduced by the error recovery technology is typically not justified in short message applications. Thus an adaptive error recovery technique is implemented in the present system.

A run length parameter l is calculated in the encoding-decoding process. The value of l is equal to the number of bits of the compressed message from the current point of a data type (any data type other than <T>) switching back to <T> to the end of the message. The system can find this value in its backward encoding process and the subscriber can find this value by keep tracking the current decoding position (the total number of bits of the information should be easily obtained by the subscriber).

First, a length threshold TL is defined. Bit padding should be implemented if $l > TL$. Otherwise, no padding is necessary. The TL was defined as 100 bits in performing the benchmark calculations for the compression rate.

It is noted that: (a) if a bit padding is implemented at the current stage, bit padding should be implemented in all the previous occurrences for the data type switching back to <T>; and (b) if a bit padding is not implemented at the current stage no bit padding should be performed in the following stages. These observations are used to simplify the implementation of the bit padding technique.

Referring to FIG. 16, the illustration shows an example of shortest path encoding of a compressed message including details such as message segmentation, shortest path data encoding, shortest path data type switch encoding, and padding in accordance with the preferred embodiment of the present invention.

The first level **1602** shows the test message "Please call BOB at 123-4567 asap. It is important." The second level **1604** shows fragmentation of the original text message into fragments separated by <cc> and/or <mini-4> tokens. The third level **1606** shows the test message encoded using the shortest path procedure. The fourth level **1608** shows the further encoding

process as data type switch tokens are identified and applied to the message data. The fifth level **1610** shows the resulting coded (compressed) information in a decimal representation, e.g., ASCII characters, phrases or the like have been replaced with their equivalent tokens and the token string has been optimized for continuations, run length occurrences, capital control, etc. The sixth level **1612** shows the padding procedure as applied to 10 bit and non- 10 bit tokens such that all blocks in the compressed message are filled upon completion of the compression process. Since all optimizations are performed on encoding (compression) and not decoding (decompression), decompression of the compressed message is much simpler than compression. Accordingly, to decompress the compressed message, one only needs to follow the top-level encoding rules in reverse.

Another feature of the present invention is its ability to mix samples for the same or dissimilar languages in order to derive an optimal token table relative to the statistical occurrence of characters, words, phrases, and other common groups of characters, numbers and symbols. The English word list disclosed is a mixture of word frequencies from different word frequency tables collected in different geographical regions. A first table was generated by collecting paging messages from Ottawa, Toronto, Montreal and Hamilton of Canada. The table comprised two data sets collected three years apart. The total period of paging messages collection is 14 days from two radio common carriers. The resultant table included 264,956 messages with 23.38% of messages in mixed case. The total number of characters in the table was 13,569,243. A second written English table was ported from the "Word Frequency Book" by John B. Carroll et. al. Finally, a third table was ported from the publication "A Word Count of Spoken English" by Davis Howes.

The tables were mixed as follows. First, items in the paging message word list are added to the mixed word list with a weight of 0.6 (60%). Second, items with a frequency greater than 0.018% (more than 500) in the written English word list are added to the mixed word list with a weight of 0.25 (25%). Last, items with frequency greater than 0.016% (more than 500) in the spoken English word list are added to the mixed word list with a weight of 0.15 (15%). From the resulting mixed table, a normalized word frequency is calculated as follows.

$$f_n = \left\lfloor \frac{f_o}{T_i} \times T_p \times F_i \right\rfloor$$

Where f_n is the updated frequency, f_o is the old frequency, T_i is the number of word count in the table (i = w for written and i = s for spoken). $T_w = 5{,}088{,}772$ and $T_s = 250{,}000$. $T_p = 2{,}165{,}082$ is the word count of the paging message word list table, and the F_i is the factor for the word table (i = p for paging, i = w for written and i = s for spoken). $F_p = 1$, $F_w = 5/12$ and $F_s = 1/4$.

Using a similar procedure, a Spanish word list is mixed. In the present invention, two tables were used for Spanish word list mixing. These tables contained paging messages and written Spanish. The first table was collected from paging messages sent in Mexico City, Mexico during a six day period from two radio common carriers. The paging messages collected were all

in upper case. The total number of messages collected was 206,543 containing 12,378,243 characters. Written Spanish was ported from two different sources, the INFOSEL and EFE corpus. The first corpus was collected from general domain Mexican news articles of about 100 MB text size. The resulting file size was 3.4 MB for the item full list with 2 MB of items frequency greater than one and 1.2 MB of items having a frequency greater than or equal to five and 62K of items frequency greater than or equal to ten.

The second corpus, EFE corpus, is a compilation of news articles of size 500 MB of widely divergent domains from the EFE satellite newswire service of Madrid, Spain. The full list is of 16.8 MB with 6.4 MB of items of frequency greater than one and 3 MB of items of frequency greater than or equal to five and 1.9 MB of items of frequency greater than or equal to ten. Only word count information was available for these statistics.

The mixing of this table was as follows. First, items in the paging message word list are added to the mixed word list with a weight of 0.6 (60%). Second, items in the INFOSEL corpus are added to the mixed word list with a weight of 0.20 (20%). Last, items in the EFE corpus are added to the mixed word list with a weight of 0.20 (20%). No spoken Spanish was added to the mixed word list.

Once the specific tables have been generated, they are applied to the text to be coded (compressed) using the processes previously discussed to generate compressed messages. It has been shown that the performance of the compression procedure described here is quite robust in a widely ranging text environments. This due mainly to the prudent selection of data types, maintaining flexible data type switching tokens and the using a powerful shortest path encoding procedure.

If say 20 tokens are removed from the token table, there is only a 0.5% negative impact on the compression rate. Thus, these 20 (or possibly a greater number) tokens can be replaced with various sets of well selected canned messages for various purposes.

The practical use of this system requires the ability to adapt the token tables to different information comprising languages, phrases, etc. To guarantee forward compatibility between dissimilar messaging units, a version control concept is introduced in the system. In this way, any subset of the tables become redefinable. Thus, new tables and old tables may coexist as defined by a version number.

The following pages represent a working specification of the preferred embodiment of the present invention for compressing and decompressing text based messages.

Token Text Compression Protocol for English and Spanish Languages

The Token Text Compression protocol is an end-application protocol used to send compressed messages over the air to wireless applications. It uses a token-based data compression techniques for lossless (i.e., the decompressed message is identical to the pre-compressed message) message transmissions. It supports 7-bit printable ASCII characters and some control characters for languages such as English and Spanish.

Token Text Compression Configuration

This protocol consists of the following fields

SIF	Dictionary ID	Data

Table 8. Token Text Compression Protocol Transaction Format

Field	M/O[1]	Definition	Size (Octets)
SIF	M	Status Information Field Value (0x6C)	1
Dictionary ID	M	Specifies the Dictionary for decompressing the message.	Expandable Integer[2]
Data	M	Data and Parameters needed to decompress the message.	Variable

1. 'M' and 'O' indicate 'Mandatory' and 'Optional', respectively.
2. Expandable Integer is defined in the Expandable Integer Appendix

There is no explicit length specification in the Data field of the Token Text Compression Protocol. The Data field contains fields of one or more data types. Each data type has its own termination rules.

Terms and Definitions

ASCII, Printable: ASCII characters from 0x20 to 0x7E.

ASCII, Control: ASCII characters from 0x00 to 0x1F.

Token: A character or a string of characters that are grouped together.

Token Table: A set of tokens that share similar characteristics. Each token in the table has a **Token ID**, a unique identification number.

Data Type: Type of data in an element in a **Token Table**. All the tokens in a token table are defined to be of the same data type. Data Types are the building blocks of the compression.

Mini-4: A data type that encodes numerals, space, and symbols that are closely associated with telephone numbers.

Mini-5: A data type that encodes words. A **Mini-5** entry may only appear next to (i.e., prior to or after) a **Mini-4** element.

Character String: A data type that is comprised of a set of individual printable characters.

Capitalization Control: A data type that is comprised of a set of tokens that control the capitalization elements in of a message. Two types of the Capitalization Control are defined: Stand Alone and Embedded.The effect of a capitalization control remains until another capitalization control token is encountered. Stand Along tokens are 10 bits long, while Embedded tokens are 2 bits long.

Fragment and Phrase: A data type that is comprised of frequently used fragments of words, complete words, phrases, and all printable ASCII.

Data Type Switch: A type of token that is used to control the switch from one data type to another.

Default Data Type: Data types that the compression algorithm utilizes when it is either at the beginning of the message or at the end of certain stages of the compression. These data types are defined to be 10-bit tokens and could be either Data Type Switch tokens or Fragment and Phrase tokens.

Non-Default Data Type: Data types that have data bit length other than 10 bits long (such as Mini-4, Mini-5, Character String, and Embedded Capitalization Control).

Leading Space: The space before a word or a character that is associated with the word or character. If multiple space characters exist, then only the last space before a word or a character is treated as a leading space.

International Characters: There are 14 characters in the ASCII table available for the display remap purpose. They are 0x23, 0x24, 0x40, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x7B, 0x7C, 0x7D, 0x7E, and 0x7F. For Spanish dictionaries, if any of these 14 characters are received, an associated font is used to display the character in stead of the standard one used in an English dictionary. Included in these 14 characters are, for example, accents and an upside-down question mark. Document 68P81139B80, "FLEX™ Character Sets", provides information on the character sets and national standards supported by FLEX™ products in a number of markets.

10-bit Tokens: Data Type Switch tokens and Fragment and Phrase tokens for which the data bit length is 10 bits long.

Token Text Compression Field Definitions

Status Information Field or 'SIF'

The Status Information Field (SIF) contains the value '0x6C'.

Dictionary ID

Dictionary ID defines the dictionary tables to be used for the decompression of the messages. Every Dictionary ID covers a set of the tables. They are Mini-4, Mini-5, Character String, Data Type Switch and Fragment and Phrase tables. In some cases, different ID share same tables. For example, the Mini-4 table is shared by both the default English and the default Spanish dictionaries.

Data

The Data field provides the data and parameters needed to decode the message. This field is comprised of tokens that are interpreted sequentially according to the compression algorithm to decompress the message.

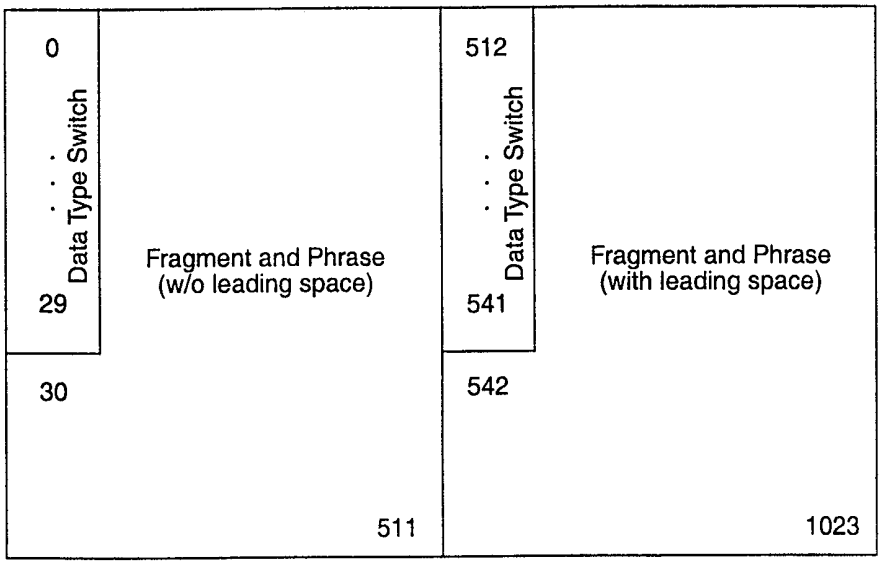
Token Text Compression Data Types

There are six data types defined in the protocol: Mini-4, Mini-5, Character String, Capitalization Control, Fragment and Phrase, and Data Type Switch. Several Capitalization Controls are also defined. The following table summarizes important data type information for the English and Spanish compression. Each will be described in later sections. The preferred tokens (which are a collection of symbols) for the default English and the default Spanish dictionaries are listed in .

Data Type	Symbol	Data Bits	Length Field	Leading Space
Mini-4	4	4	(4+1)	-
Mini-5	5	6	(1+1)	yes
Character String	c	6	(2+1)	-
Fragment and Phrase	-	10	-	yes
Data Type Switch	-	10	-	-
Capitalization Control-Embedded	0	2	-	-
Capitalization Control-All Cap	6	10	-	-
Capitalization Control-All Low	7	10	-	-
Capitalization Control-Cap First	8	10	-	-
Capitalization Control-Reserved	9	10	-	-

Table 9. Summary of Data Types for English and Spanish Compression

Fragment and Phrase data type and Data Type Switch data type share the same range of token identification numbers: from 0 to 1023. Fragment and Phrase tokens from 30 to 511 and from 542 to 1023 are identical except that the later set has a leading space. The following map shows the locations of each:



Mini-4

There are 16 elements in the Mini-4 token table. One or more Mini-4 token entries are specified using the following format:

Length Field (length [+ continuation bit])	Data

The Data field consists of one of more Mini-4 tokens, as specified by the Length Field. The Length Field has a (4+1) format. The 4 is used to indicate the data count in the Data field (i.e., the number of Mini-4 tokens) immediately following the Length Field, and the 1 is an optional continuation bit. The Length Field and Data pattern repeats if the continuation bit is present and set. The possible Length Fields are listed in Table 10,

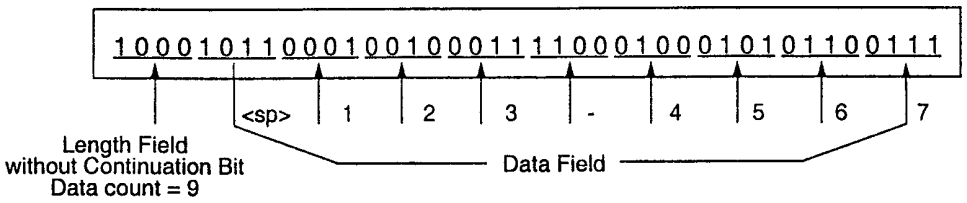
length bit-pattern	continua-tion bit	data count
0000	-	1
0001	-	2
0010	-	3
0011	-	4
0100	-	5
0101	-	6
0110	-	7
0111	-	8
1000	-	9
1001	-	10
1010	-	11
1011	-	12
1100	-	13
1101	-	14
1110	-	15
1111	0	16
1111	1	17+

Table 10. Mini-4 Length Field

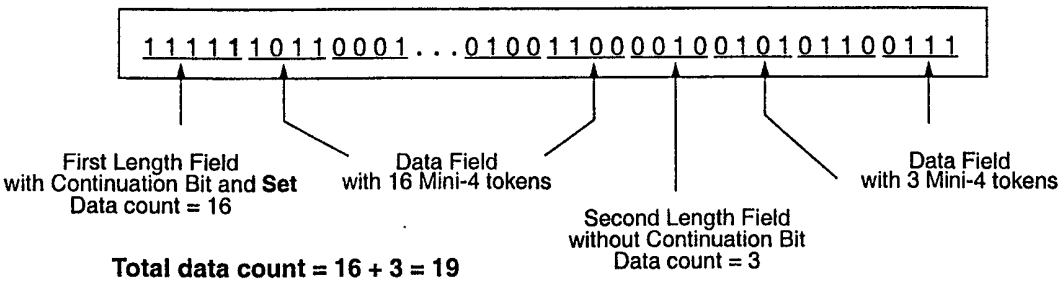
Each Mini-4 token is identified by a 4-bit identification number in the range of 0 to 15. The Mini-4 tokens for the default English and the default Spanish dictionaries are listed in Table 14.

Example of Mini-4 in a compressed data format:

Case 1: Data count less than 16 (default English dictionary)



Case 2: Data count more than 16 (default English dictionary)



Total data count = 16 + 3 = 19

Mini-5

There are 32 elements in the Mini-5 token table. One or more Mini-5 token entries are specified using the following format:

Length Field (length [+ continuation bit])	Data

The Data field consists of one of more Mini-5 tokens, as specified by the Length Field. The Length Field has a (1+1) format. The first 1 is used to indicate the data count in the Data field (i.e., the number of Mini-5 tokens) immediately following the Length Field, and the second 1 is an optional continuation bit. The Length Field and Data pattern repeats if the continuation bit is present and set. The possible Length Fields are listed in Table 11.

length bit-pattern	continuation bit	data count
0	-	1
1	0	2
1	1	3+

Table 11. Mini-5 Length Field

Each Mini-5 token is identified by a 6 bit value: a 5-bit identification number in the range of 0 to 31, and a 1-bit leading space indicator. The leading space indicator bit is the most significant bit of the 6 bits. If the leading indicator bit is set, it indicates that a Leading Space is present. The Mini-5 tokens for the default English and the default Spanish dictionaries are listed in Tables 17 and 18.

Character String

There are 64 elements in the Character String token table. One or more Character String token entries are specified using the following format:

Length Field (length [+ continuation bit])	Data

The Data field consists of one of more Character String tokens, as specified by the Length Field. The Length Field has a (2+1) format. The 2 is used to indicate the data count in the Data field (i.e., the number of Character String tokens) immediately following the Length Field, and the 1 is an optional continuation bit. The Length Field and Data pattern repeats if the continuation bit is present and set. The possible Length Fields are listed in Table 12.

length bit-pattern	continuation bit	data count
00	-	1
01	-	2
10	-	3
11	0	4
11	1	5+

Table 12. Character String Length Field

Each Character String token is identified by a 6-bit identification number in the range of 0 to 63. The Character String tokens for the default English and the default Spanish dictionaries are listed in Table 19.

Fragment and Phrase

There are 512 elements in the Fragment and Phrase token table. The Fragment and Phase table contains the most commonly used fragments, words and phrases for the dictionary. This data type (together with Data Type Switch) defines the default data type. A Fragment and Phrase token is identified by a 10 bit value: a 9-bit identification number in the range of 30 to 511 (the range of 0 to 31 is used for data type switching), and a 1-bit leading space indicator. The leading space indicator bit is the most significant bit of the 10 bits. If the leading indicator bit is set, it indicates that a Leading Space is present. The Fragment and Phrase tokens for the default English and the default Spanish dictionaries are listed in .

Capitalization Control

Two types of capitalization controls are defined: Stand Alone and Embedded. They are identical in function. Stand Alone capitalization control tokens are part of the Data Type Switch tokens and are 10 bits long. The Embedded capitalization control tokens are 2 bits long. The Embedded Capitalization Control tokens are represented by the symbol '0' in the Data Type Switch token strings. It identifies the relative location of the actual 2-bit data of the Capitalization Control token. The default case of a message is upper case. Thus, if a message is all in upper case, there is no need for a capitalization control.

There are three types of capitalization controls for both Stand Alone and Embedded tokens. They are All Cap, All Low, and Cap First.

All Cap

All letters from here on will be transformed into upper case. The effect will be terminated when another capitalization control token is encountered.

All Low

All letters from here on will be transformed into lower case. The effect will be terminated when another capitalization control token is encountered.

Cap First

The next letter encountered is capitalized. The capitalization flag is changed to lower case after the capitalization. If the character immediately after the Cap First token is not a letter, Cap First token remains effective until it encounters a letter and capitalizes it. The IDs for the capitalization control tokens are listed inTable 13.

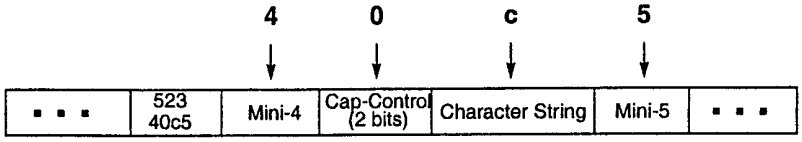
	Stand Alone (10 bits)	Embedded (2 bits)
All Cap	2	0
All Low	3	1
Cap First	4	2
(reserved)	5	3

Table 13. Capitalization Control ID

Data Type Switch

Data Type Switch tokens are used to group non-default data types. There are 60 Data Type Switch tokens. The possible non-default data types for a Data Type Switch are Mini-4(4), Mini-5(5), Character String(c), and Embedded Capitalization Control(0). Data Type Switch tokens are 10 bits long. The Data Type Switch tokens for the default English and the default Spanish dictionaries are listed in Tables 20 and 21.

Example: default English dictionary, ID=523, value in table=40c5



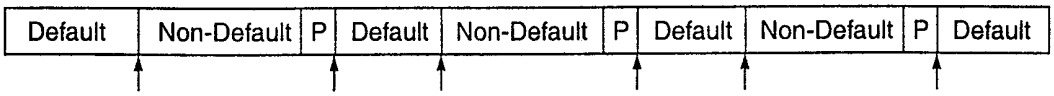
When decompressing the message using the default English dictionary, after the identification of the 10-bit token with ID 523 as a Data Type Switch token, the Data Type Switch Token table is used. The table entry for ID 523 is 40c5. This implies that the following data types are Mini-4, Embedded Capitalization Control, Character String, and Mini-5. Only the Embedded Capitalization Control has a finite bit length, i.e., 2 bits. The lengths for the other types vary depending on the Length Field for each type. The type of the Capitalization Control is unknown until the 2 bits are read. Note that each field has its own termination rule. When decompressing, after the processing of the last data type (in this example, Mini-5) is completed, the next 10 bits of data are interpreted to continue the process of decompressing the message.

Token Text Compression Data Padding

Compression Padding

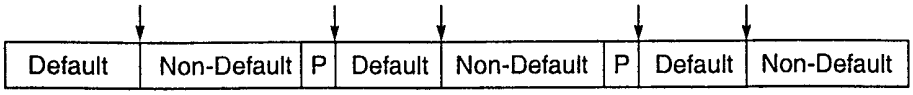
All data types are classified into two types: Default Data Type and Non-Default Data Type. The default Data Types are Data Type Switch, and Fragment and Phrase. The non-Default Data Types are Mini-4, Mini-5, and Character String. The data of Default Data Type are 10 bits long. All the Non-Default Data Types are padded with '0's to the next multiple of 10-bit boundary if not already on the boundary, except for the last group. The following is a graphical description of the padding for the Token Text Compression.

Case 1:



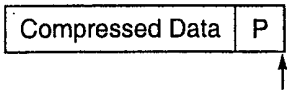
Case 2:

Multiple of 10-bit boundaries



Data Alignment Padding

After the compression, the output is padded to be an integral number of octets. The data is padded with '0's to the next multiple of 8-bit boundary if not already on the boundary.



Multiple of 8-bit boundary

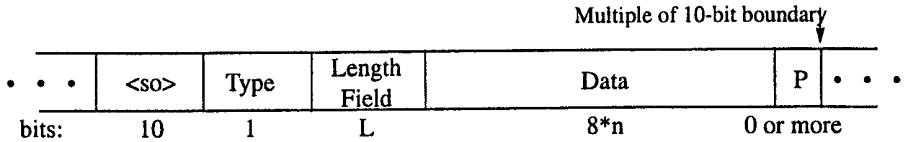
Token Text Compression Special Data Handling

Token Text Compression may be used to specify portions of the data which are not compressed. To specify such data, a special case mode is used. The following rules are used to specify the data:

- <so> is used for the mode change of from compressed to special case mode.
- Compression Padding rule applies to data after <so>.
- The capitalization control flag does not reset after the handling of the special data section.

Note that the default mode for Token Text Compression is compressed mode.

The format of the fields are:



<so> : a 10-bit indicator for the starting of Special Data Handling section of the Token Text Compression.
Type : a 1-bit field used to interpret the Length Field.
Length Field : a length indication.
Data : Data octets. The Length Field specifies the number of octets, n, in the Data field.
P : Compression padding, 0 or more bits to the next multiple of 10-bit boundary.

Type	Length Field
0	1+1
1	Expandable Integer

Table 14. Special Case Mode: Type & Length Field Definition

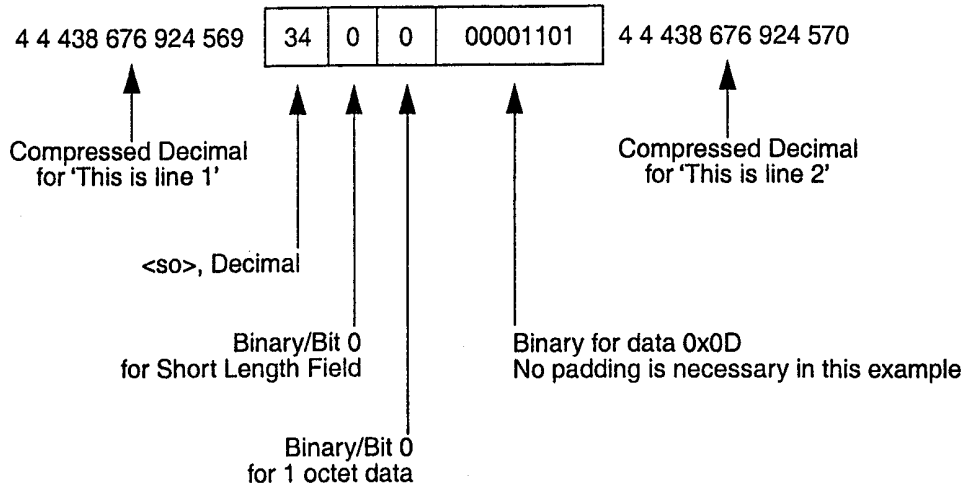
If the Type is 0, then the first 1 is used to indicate the number of octets immediately following the Length Field, and the second 1 is an optional continuation bit. The Length Field and Data pattern repeats if the continuation bit is present and set. The possible Length Fields are listed in Table 15.

length bit-pattern	continuation bit	data count
0	-	1
1	0	2
1	1	3+

Table 15. Special Case Mode: Length Field

Example using the default English dictionary:

ASCII 0x0D, <cr>, is not in any table but is part of the message as: This is line 1<cr>This is line 2

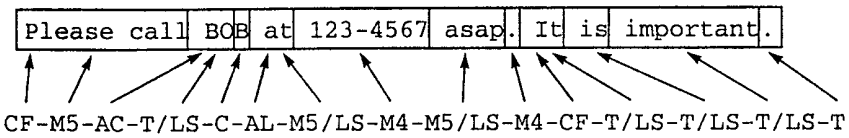


Token Compression Example

Original Message:

Please call BOB at 123-4567 asap. It is important.

The message is to be compressed using the default English dictionary. The message is broken into (NOTE: one of many possibilities)



Compressed Message in decimal before padding:

18 2 0 30 0 634 15 0 37 1 13 0 40 8 10 1 2 3 12 4 5 6 7 28 0 48 0 13 2 677 676 1019 54

Legend:

- CF : Cap First
- M5 : Mini-5
- AC : All Cap
- T/LS : Fragment with Leading Space
- C : Character String
- AL : All Low
- M4 : Mini-4
- M5/LS : Mini-5 with Leading Space
- T : Fragment

The meaning of the tokens and the conversion of decimal tokens to binary output with padding for the compression and data alignment are summarized in the following table:

Data/Decimal	Data Type	Description	Decompressed Message	bit length	binary/bits
18	Data Type Switch	050		10	0000010010
2	Capitalization Control	Cap First		2	10
0	Mini-5	Length Field		1	0
30	Mini-5	Data	**Please call**	6	011110
0	Capitalization Control	All Cap		2	00
padding1				9	000000000
634	Fragment & Phrase	Data with Leading Space	**<sp>BO**	10	1001111010
15	Data Type Switch	c0		10	0000001111
0	Character String	Length Field		2	00
37	Character String	Data	**B**	6	100101
1	Capitalization Control	All Low		2	01
padding1				0	
13	Data Type Switch	54		10	0000001101
0	Mini-5	Length Field		1	0
40	Mini-5	Data with Leading Space	**<sp>at**	6	101000
8	Mini-4	Length Field		4	1000
10	Mini-4	Data	**<sp>**	4	1010
1	Mini-4	Data	**1**	4	0001
2	Mini-4	Data	**2**	4	0010
3	Mini-4	Data	**3**	4	0011
12	Mini-4	Data	**-**	4	1100
4	Mini-4	Data	**4**	4	0100
5	Mini-4	Data	**5**	4	0101
6	Mini-4	Data	**6**	4	0110
7	Mini-4	Data	**7**	4	0111
padding1				3	000
28	Data Type Switch	540		10	0000011100
0	Mini-5	Length Field		1	0
48	Mini-5	Data with Leading Space	**<sp>asap**	6	110000
0	Mini-4	Length Field		4	0000
13	Mini-4	Data	**.**	4	1101
2	Capitalization Control	Cap First		2	10
padding1				3	000
677	Fragment & Phrase	Data with Leading Space	**<sp>It**	10	1010100101
676	Fragment & Phrase	Data with Leading Space	**<sp>is**	10	1010100100
1019	Fragment & Phrase	Data with Leading Space	**<sp>important**	10	1111111011
54	Fragment & Phrase	Data	**.**	10	0000110110
padding2				2	00

NOTE:

padding1 - Compression Padding
padding2 - Data Alignment Padding

Concatenate the binary data at the right-most column of the above table, split the binary string into octets, and the result is listed in the following table:

Binary	0000 0100	1010 0011	1100 0000	0000 0010	0111 1010	0000 0011	1100 1001	0101 0000
Hexadecimal	04	A3	C0	02	7A	03	C9	50
Binary	0011 0101	0100 0100	0101 0000	1001 0001	1110 0010	0010 1011	0011 1000	0000 0111
Hexadecimal	35	44	50	91	E2	2B	38	07
Binary	0001 1000	0000 0110	1100 0010	1010 0101	1010 1001	0011 1111	1011 0000	1101 1000
Hexadecimal	18	06	C2	A5	A9	3F	B0	D8

The final data of this compression in hexadecimal is:

04 A3 C0 02 7A 03 C9 50 35 44 50 91 E2 2B 38 07 18 06 C2 A5 A9 3F B0 D8

Add the Token Text Compression SIF, a DID (assuming it is 0x00), the final data becomes

SIF	Dictionary ID	Data
6C	00	04 A3 C0 02 7A 03 C9 50 35 44 50 91 E2 2B 38 07 18 06 C2 A5 A9 3F B0 D8

Mini-4 Token Table

ID	Token
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	<sp>
11	,
12	-
13	.
14	/
15	:

Table 16. Default English and Spanish Mini-4

Mini-5 Token Tables

ID	Token	ID	Token
0	#	16	asap
1	$	17	call
2	(	18	pls.
3	)	19	after
4	a	20	today
5	f	21	office
6	p	22	please
7	am	23	pls cl
8	at	24	thanks
9	pm	25	at home
10	re	26	pls call
11	dir	27	tomorrow
12	ext	28	go direct
13	for	29	pls. call
14	pls	30	please call
15	re:	31	returned your call

Table 17. Default English Mini-5

ID	Token	ID	Token
0	*	16	con la
1	[	17	de las
2	]	18	espero
3	y	19	com. al
4	].	20	urgente
5	al	21	urgente.
6	de	22	hasta las
7	com	23	urgente al
8	con	24	comunicarse
9	ext	25	antes de las
10	tel	26	manana a las
11	para	27	comunicate al
12	a las	28	comunicarse al
13	com al	29	favor comunicarse al
14	com.al	30	favor de comunicarse
15	con el	31	este radio. comunicarse

Table 18. Default Spanish Mini-5

Character String Token Table

ID	Token	ID	Token	ID	Token	ID	Token
0	<sp>	16	1	32	[	48	m
1	!	17	2	33	]	49	n
2	"	18	3	34	^	50	o
3	#	19	4	35	_	51	p
4	%	20	5	36	a	52	q
5	&	21	6	37	b	53	r
6	'	22	7	38	c	54	s
7	(	23	8	39	d	55	t
8	)	24	9	40	e	56	u
9	*	25	:	41	f	57	v
10	+	26	;	42	g	58	w
11	,	27	<	43	h	59	x
12	-	28	=	44	i	60	y
13	.	29	>	45	j	61	z
14	/	30	?	46	k	62	{
15	0	31	@	47	l	63	}

Table 19. Default English and Spanish Character String

Data Type Switch Token Tables

ID	Token	ID	Token	ID	Token	ID	Token
0	4	15	c0	512	54c	527	5454
1	5	16	c4	513	5c0	528	54c4
2	6	17	c5	514	5c4	529	c40c
3	7	18	050	515	5c5	530	c450
4	8	19	054	516	c40	531	c454
5	9	20	0c4	517	c45	532	c4c4
6	c	21	405	518	c4c	533	c545
7	05	22	40c	519	c50	534	c5c4
8	0c	23	450	520	c54	535	45454
9	40	24	454	521	c5c	536	54545
10	45	25	45c	522	40c4	537	c4545
11	4c	26	4c4	523	40c5	538	c5454
12	50	27	4c5	524	4545	539	0c40c4
13	54	28	540	525	4c4c	540	545454
14	5c	29	545	526	4c54	541	c54545

Table 20. Default English Data Type Switch

ID	Token	ID	Token	ID	Token	ID	Token
0	4	15	c0	512	54c	527	5c45
1	5	16	c4	513	5c0	528	5c54
2	6	17	c5	514	5c4	529	c450
3	7	18	050	515	5c5	530	c454
4	8	19	054	516	c40	531	c4c4
5	9	20	0c4	517	c45	532	c545
6	c	21	405	518	c4c	533	c54c
7	05	22	40c	519	c50	534	c5c4
8	0c	23	450	520	c54	535	45454
9	40	24	454	521	c5c	536	54545
10	45	25	45c	522	4545	537	c4545
11	4c	26	4c4	523	4c4c	538	c5454
12	50	27	4c5	524	4c54	539	454545
13	54	28	540	525	5454	540	4c4c4c
14	5c	29	545	526	54c4	541	c54545

Table 21. Default Spanish Data Type Switch

Fragment and Phrase Token Tables

ID	Token	ID	Token	ID	Token	ID	Token	ID	Token	ID	Token	ID	Token	ID	Token
30	<nul>	91	m	152	ge	213	se	274	eri	335	now	396	cher	457	frank
31	<eot>	92	n	153	go	214	sh	275	err	336	oct	397	come	458	front
32	<bel>	93	o	154	ha	215	si	276	est	337	off	398	conf	459	going
33	<nl>	94	p	155	he	216	so	277	eve	338	one	399	cont	460	group
34	<so>	95	q	156	hi	217	sp	278	fax	339	out	400	dave	461	house
35	<si>	96	r	157	ho	218	ss	279	feb	340	par	401	down	462	inter
36	<esc>	97	s	158	ie	219	st	280	fer	341	pat	402	from	463	linda
37	<fs>	98	t	159	if	220	su	281	for	342	per	403	have	464	needs
38	<gs>	99	u	160	il	221	ta	282	fri	343	pla	404	here	465	offer
39	<rs>	100	v	161	im	222	te	283	gar	344	pls	405	hill	466	peter
40	<sp>	101	w	162	in	223	th	284	ger	345	pro	406	home	467	phone
41	!	102	x	163	ir	224	ti	285	get	346	que	407	john	468	ridge
42	"	103	y	164	is	225	to	286	gra	347	rea	408	just	469	steve
43	#	104	z	165	it	226	tr	287	gre	348	rec	409	king	470	store
44	$	105	{	166	ja	227	tt	288	ham	349	reg	410	know	471	thank
45	%	106	|	167	jo	228	ty	289	har	350	ren	411	land	472	there
46	&	107	}	168	ka	229	un	290	has	351	res	412	line	473	today
47	'	108	~	169	ke	230	up	291	hea	352	ric	413	lock	474	at the
48	(	109	's	170	ki	231	ur	292	her	353	rie	414	mail	475	before
49	)	110	ac	171	la	232	us	293	him	354	ron	415	mari	476	called
50	*	111	ad	172	ld	233	ut	294	his	355	ros	416	mark	477	cancel
51	+	112	al	173	le	234	wa	295	ian	356	rus	417	mary	478	george
52	,	113	am	174	li	235	wi	296	ice	357	san	418	ment	479	hookup
53	-	114	an	175	ll	236	age	297	ill	358	sat	419	mike	480	in the
54	.	115	ar	176	lo	237	ale	298	ine	359	see	420	need	481	ington
55	/	116	as	177	ly	238	all	299	ing	360	sep	421	ning	482	number
56	0	117	at	178	ma	239	and	300	ins	361	sha	422	open	483	office
57	1	118	av	179	mc	240	ann	301	int	362	she	423	page	484	please
58	2	119	ba	180	me	241	app	302	ion	363	son	424	paul	485	return
59	3	120	be	181	mi	242	apr	303	ive	364	sta	425	pick	486	robert
60	4	121	bi	182	mo	243	ard	304	jan	365	ste	426	port	487	server
61	5	122	bo	183	mr	244	are	305	jul	366	sti	427	real	488	system
62	6	123	br	184	na	245	arl	306	jun	367	sun	428	rece	489	thanks
63	7	124	bu	185	nd	246	aug	307	key	368	tal	429	righ	490	urgent
64	8	125	ca	186	ne	247	ave	308	lan	369	tel	430	room	491	at home
65	9	126	ce	187	ng	248	bar	309	lea	370	ter	431	ship	492	call me
66	:	127	ch	188	ni	249	bel	310	len	371	the	432	show	493	confirm
67	;	128	ci	189	no	250	ber	311	les	372	thu	433	sign	494	http://
68	<	129	ck	190	ns	251	ble	312	ley	373	tin	434	ston	495	meeting
69	=	130	cl	191	nt	252	box	313	lin	374	ton	435	test	496	message
70	>	131	co	192	of	253	bra	314	lis	375	tra	436	that	497	morning
71	?	132	cr	193	on	254	bro	315	lle	376	tue	437	ther	498	problem
72	@	133	ct	194	op	255	bur	316	loo	377	use	438	this	499	richard
73	[	134	da	195	or	256	can	317	man	378	van	439	time	500	service
74	\	135	de	196	os	257	car	318	mar	379	ver	440	ting	501	tonight
75	]	136	di	197	pa	258	cha	319	may	380	way	441	tion	502	call the
76	^	137	do	198	pe	259	che	320	mer	381	wed	442	when	503	reminder
77	_	138	dr	199	pm	260	col	321	mil	382	wel	443	will	504	tomorrow
78	`	139	du	200	po	261	com	322	min	383	wit	444	wood	505	at office
79	a	140	dy	201	pr	262	con	323	mis	384	www	445	work	506	go direct
80	b	141	ea	202	pu	263	cor	324	mon	385	you	446	your	507	important
81	c	142	ed	203	ra	264	cou	325	mor	386	ance	447	about	508	call office
82	d	143	el	204	rd	265	day	326	mrs	387	andr	448	after	509	at the office
83	e	144	en	205	re	266	dec	327	n't	388	anne	449	ation	510	in the office
84	f	145	er	206	ri	267	del	328	net	389	appt	450	brian	511	call the office
85	g	146	es	207	ro	268	den	329	new	390	asap	451	carol		
86	h	147	et	208	rs	269	der	330	nge	391	back	452	chris		
87	i	148	fa	209	rt	270	don	331	nic	392	bank	453	david		
88	j	149	fi	210	ry	271	ell	332	nne	393	bill	454	field		
89	k	150	fr	211	sa	272	end	333	not	394	call	455	first		
90	l	151	ga	212	sc	273	ent	334	nov	395	cent	456	floor		

Table 22. Default English Fragment and Phrase

ID	Token	ID	Token	ID	Token	ID	Token	ID	Token	ID	Token	ID	Token	ID	Token
30	<nul>	91	m	152	lo	213	dia	274	ria	335	luis	396	tiene	457	mensaje
31	<eot>	92	n	153	lu	214	dor	275	rio	336	mama	397	adrian	458	mensual
32	<bel>	93	o	154	ma	215	dra	276	ros	337	marc	398	arturo	459	oficina
33	<nl>	94	p	155	me	216	eia	277	sal	338	mari	399	asunto	460	posible
34	<so>	95	q	156	mi	217	eli	278	san	339	mayo	400	buenos	461	recoger
35	<si>	96	r	157	mo	218	end	279	son	340	medi	401	carlos	462	recuerd
36	<esc>	97	s	158	mu	219	era	280	sra	341	oche	402	carmen	463	reporta
37	<fs>	98	t	159	na	220	ero	281	sta	342	ofna	403	cheque	464	ricardo
38	<rs>	99	u	160	ne	221	err	282	ste	343	pago	404	ciones	465	saludos
39	<gs>	100	v	161	ni	222	esp	283	sus	344	papa	405	con el	466	sanchez
40	<sp>	101	w	162	no	223	est	284	tan	345	para	406	contra	467	textual
41	!	102	x	163	ol	224	fav	285	tar	346	port	407	cuando	468	urgente
42	"	103	y	164	on	225	feb	286	tel	347	pres	408	cuanto	469	comunica
43	#	104	z	165	or	226	gar	287	ten	348	pued	409	daniel	470	confirma
44	$	105	{	166	os	227	gas	288	ter	349	raul	410	de com	471	despacho
45	%	106	|	167	pa	228	gra	289	tes	350	reci	411	entreg	472	encuentr
46	&	107	}	168	pe	229	hay	290	tor	351	sali	412	espera	473	favor de
47	'	108	~	169	pi	230	hij	291	tos	352	segu	413	espero	474	gonzalez
48	(	109	al	170	po	231	hoy	292	tra	353	sept	414	esposa	475	hospital
49	)	110	am	171	ra	232	ica	293	una	354	todo	415	garcia	476	mauricio
50	*	111	an	172	re	233	ico	294	uni	355	urge	416	hablar	477	necesita
51	+	112	ar	173	ri	234	ida	295	urg	356	abril	417	herman	478	necesito
52	,	113	as	174	ro	235	ido	296	www	357	acion	418	javier	479	patricia
53	-	114	at	175	ru	236	ill	297	a la	358	angel	419	llamar	480	problema
54	.	115	ba	176	sa	237	ina	298	ador	359	antes	420	llegue	481	servicio
55	/	116	be	177	se	238	ing	299	alle	360	berto	421	manana	482	a su casa
56	0	117	bo	178	si	239	ist	300	amor	361	david	422	manuel	483	alejandro
57	1	118	ca	179	so	240	ita	301	amos	362	donde	423	martha	484	comunique
58	2	119	ce	180	sr	241	las	302	andr	363	encia	424	martin	485	encuentra
59	3	120	ch	181	su	242	les	303	aqui	364	entes	425	miento	486	esperando
60	4	121	ci	182	ta	243	lic	304	atte	365	ernes	426	miguel	487	francisco
61	5	122	co	183	te	244	lla	305	avis	366	estoy	427	porque	488	hernandez
62	6	123	cu	184	ti	245	los	306	beso	367	favor	428	prueba	489	noviembre
63	7	124	da	185	to	246	man	307	bien	368	gomez	429	puedas	490	por favor
64	8	125	de	186	tu	247	mar	308	buen	369	hablo	430	puedes	491	rodriguez
65	9	126	di	187	un	248	mas	309	casa	370	hasta	431	quiere	492	comunicate
66	:	127	do	188	ur	249	men	310	cast	371	iente	432	quiero	493	comuniques
67	;	128	dr	189	va	250	mer	311	cion	372	inter	433	rafael	494	suspension
68	<	129	el	190	aca	251	mex	312	cita	373	jaime	434	sergio	495	comunicarse
69	=	130	en	191	ada	252	min	313	como	374	jesus	435	trabaj	496	comunicarte
70	>	131	er	192	ado	253	mon	314	comp	375	jorge	436	alfonso	497	consultorio
71	?	132	es	193	ale	254	ndo	315	cons	376	julio	437	alfredo	498	de parte de
72	@	133	fa	194	ame	255	nos	316	cont	377	junio	438	antonio	499	despertador
73	[	134	fe	195	ana	256	nov	317	dias	378	laura	439	celular	500	presentarse
74	\	135	ga	196	ano	257	nte	318	ecto	379	llamo	440	claudia	501	a la oficina
75	]	136	go	197	arq	258	oct	319	edad	380	llega	441	clinica	502	favor de com
76	^	137	gu	198	bre	259	pac	320	emos	381	llego	442	conmigo	503	comunicarse a
77	_	138	ha	199	cam	260	pas	321	enta	382	lopez	443	contigo	504	comunicarse al
78	`	139	hs	200	can	261	ped	322	ente	383	manda	444	despues	505	de comunicarse
79	a	140	ia	201	car	262	per	323	entr	384	mando	445	eduardo	506	comunicarse con
80	b	141	ic	202	cha	263	por	324	eron	385	marzo	446	enrique	507	favor comunicarse
81	c	142	il	203	cho	264	pre	325	esta	386	mente	447	estamos	508	favor de comunicarse
82	d	143	in	204	cia	265	pro	326	este	387	mento	448	fernand	509	por favor comunicate
83	e	144	io	205	cio	266	pue	327	habl	388	mismo	449	gabriel	510	comunicarse a su casa
84	f	145	ir	206	com	267	que	328	hace	389	mucho	450	gerardo	511	favor de comunicarse con
85	g	146	is	207	con	268	qui	329	hola	390	olvid	451	gracias		
86	h	147	ja	208	cor	269	ran	330	hora	391	pasar	452	hablame		
87	i	148	la	209	dad	270	rec	331	jose	392	saber	453	http://		
88	j	149	le	210	del	271	reg	332	juan	393	srita	454	informa		
89	k	150	li	211	der	272	ren	333	lave	394	tarde	455	llamada		
90	l	151	il	212	des	273	res	334	llam	395	tengo	456	llamame		

Table 23. Default Spanish Fragment and Phrase

In accordance with the preferred embodiment of the present invention, only one bit of information is required on the presentation layer to specify whether the message is compressed, while m bits are required for the specification of the version number. The selection of the number m depends on whether it is a system specific or a customer specific version. A system specific version number requires uniqueness within the whole system while the customer specific version number requires the uniqueness in regards to that customer only. The former requires more bits while the latter less. It is suggested that customer compression version mapping should be part of the customer's profile at a radio common carrier or the like.

In the preceding examples, the messaging information is composed using conventional computers and data structures, and the message is compressed using the unique procedure disclosed herein. After each message is compressed, it may be sent like a normal paging message to the paging system via the public switched telephone network or the like.

One of ordinary skill in the art will appreciate that the preceding discussion regarding the claimed invention in not meant to limit the system to a particular transport protocol, wireless media, compression scheme, or physical communication device. Consequently, the claimed invention and other variations made possible by the teachings herein represent only a few select ways that a secure messaging system for communicating information can be implemented using the unique principles taught in the present invention.

It is in the preceding spirit that we claim the following as our invention:

CLAIMS

1. A reduced overhead text messaging system comprising:

a lossless compression engine that operates on an original message to generate a compressed message including at least one of a data type switch token and a fragment and phrase token.

2. The reduced overhead text messaging system according to claim 1 wherein the compressed message is encoded using a shortest path context-dependent cost matrix.

3. The reduced overhead text messaging system according to claim 2 wherein the original message is fragmented before it is encoded using the shortest path context-dependent cost matrix.

4. The reduced overhead text messaging system according to claim 1 wherein the compressed message comprises at least one of a mini-4 token, a mini-5 token, a character string token, the fragment and phrase token, the data type switch token, and a capitalization control token.

5. The reduced overhead text messaging system according to claim 4 wherein the mini-4 token is a member of a table of mini-4 tokens each representing a compressed element.

6. The reduced overhead text messaging system according to claim 5 wherein the mini-4 token comprises 4 bits.

7. The reduced overhead text messaging system according to claim 6 wherein the table of mini-4 tokens comprises at least 16 elements.

8. The reduced overhead text messaging system according to claim 5 wherein the table of mini-4 tokens comprises less than 16 elements.

9. The reduced overhead text messaging system according to claim 5 wherein the table of mini-4 tokens comprises more than 16 elements.

10. The reduced overhead text messaging system according to claim 7 wherein the table of mini-4 tokens comprises 16 index-token pairs of:

0, <0>; 1, <1>; 2, <2>; 3, <3>; 4, <4>; 5, <5>; 6, <6>; 7, <7>; 8, <8>; 9, <9>; 10, <space>; 11, <comma>; 12, <hyphen>; 13, <period>; 14, <forward-slash>; and 15, <colon>.

11. The reduced overhead text messaging system according to claim 4 wherein the mini-5 token is a member of a table of mini-5 tokens each representing a compressed element.

12. The reduced overhead text messaging system according to claim 11 wherein the mini-5 token comprises 6 bits.

13. The reduced overhead text messaging system according to claim 12 wherein the table of mini-5 tokens comprises at least 32 elements.

14. The reduced overhead text messaging system according to claim 11 wherein the table of mini-5 tokens comprises less than 32 elements.

15. The reduced overhead text messaging system according to claim 11 wherein the table of mini-5 tokens comprises more than 32 elements.

16. The reduced overhead text messaging system according to claim 13 wherein the table of mini-5 tokens comprises 32 index-token pairs of:

0, <number-symbol>; 1, <dollar-sign>; 2, <left-parenthesis>; 3, <right-parenthesis>; 4, <a>; 5, <f>; 6, <p>; 7, <am>; 8, <at>; 9, <pm>; 10, <re>; 11, <dir>; 12, <ext>; 13, <for>; 14, <pls>; 15, <re:>; 16, <asap>; 17, <call>; 18, <pls.>; 19, <after>; 20, <today>; 21, <office>; 22, <please>; 23, <pls cl>; 24, <thanks>; 25, <at home>; 26, <pls call>; 27, <tomorrow>; 28, <go direct>; 29, <pls. call>; 30, <please call>; and 31, <returned your call>.

17. The reduced overhead text messaging system according to claim 4 wherein the character string token is a member of a table of character string tokens each representing a compressed element.

18. The reduced overhead text messaging system according to claim 17 wherein the character string token comprises 6 bits.

19. The reduced overhead text messaging system according to claim 18 wherein the table of character string tokens comprises at least 64 elements.

20. The reduced overhead text messaging system according to claim 17 wherein the table of character string tokens comprises less than 64 elements.

21. The reduced overhead text messaging system according to claim 17 wherein the table of character string tokens comprises more than 64 elements.

22. The reduced overhead text messaging system according to claim 19 wherein the table of character string tokens comprises 64 index-token pairs of:

0, <space>; 1, <exclamation mark>; 2, <double quote>; 3, <number sign>; 4, <percent sign>; 5, <ampersand>; 6, <apostrophe>; 7, <left parenthesis>; 8, <right parenthesis>; 9, <asterisk>; 10, <plus sign>; 11, <comma>; 12, <hyphen or minus sign>; 13, <period>; 14, <forward slash>; 15, <0>; 16, <1>; 17, <2>; 18, <3>; 19, <4>; 20, <5>; 21, <6>; 22, <7>; 23, <8>; 24, <9>; 25, <colon>; 26, <semicolon>; 27, <less than symbol>; 28, <equal symbol>; 29, <greater than symbol>; 30, <question mark>; 31, <at symbol>; 32, <left square bracket>; 33, <right square bracket>; 34, <caret>; 35, <underscore>; 36, <a>; 37, <b>; 38, <c>; 39, <d>; 40, <e>; 41, <f>; 42, <g>; 43, <h>; 44, <i>; 45, <j>; 46, <k>; 47, <l>; 48, <m>; 49, <n>; 50, <o>; 51, <p>; 52, <q>; 53, <r>; 54, <s>; 55, <t>; 56, <u>; 57, <v>; 58, <w>; 59, <x>; 60, <y>; 61, <z>; 62, <left curly bracket>; and 63, <right curly bracket>.

23. The reduced overhead text messaging system according to claim 4 wherein the data type switch token is a member of a table of data type switch tokens each representing a compressed element.

24. The reduced overhead text messaging system according to claim 23 wherein the data type switch token comprises 10 bits.

25. The reduced overhead text messaging system according to claim 24 wherein the table of data type switch tokens comprises at least 60 elements.

26. The reduced overhead text messaging system according to claim 23 wherein the table of data type switch tokens comprises less than 60 elements.

27. The reduced overhead text messaging system according to claim 23 wherein the table of data type switch tokens comprises more than 60 elements.

28. The reduced overhead text messaging system according to claim 25 wherein the table of data type switch tokens comprises 60 index-token pairs of:

0, <4>; 1, <5>; 2, <6>; 3, <7>; 4, <8>; 5, <9>; 6, <c>; 7, <05>; 8, <0c>; 9, <40>; 10, <45>; 11, <4c>; 12, <50>; 13, <54>; 14, <5c>; 15, <c0>; 16, <c4>; 17, <c5>; 18, <050>; 19,

<054>; 20, <0c4>; 21, <405>; 22, <40c>; 23, <450>; 24, <454>; 25, <45c>; 26, <4c4>; 27, <4c5>; 28, <540>; 29, <545>; 512, <54c>; 513, <5c0>; 514, <5c4>; 515, <5c5>; 516, <c40>; 517, <c45>; 518, <c4c>; 519, <c50>; 520, <c54>; 521, <c5c>; 522, <40c4>; 523, <40c5>; 524, <4545>; 525, <4c4c>; 526, <4c54>; 527, <5454>; 528, <54c4>; 529, <c40c>; 530, <c450>; 531, <c454>; 532, <c4c4>; 533, <c545>; 534, <c5c4>; 535, <45454>; 536, <54545>; 537, <c4545>; 538, <c5454>; 539, <0c40c4>; 540, <545454>; and 541, <c54545>.

29. The reduced overhead text messaging system according to claim 4 wherein the capitalization control token is a member of a table of capitalization control tokens each representing a compressed element.

30. The reduced overhead text messaging system according to claim 29 wherein the capitalization control token comprises 2 bits when the capitalization control token is embedded in a selected data type switch token.

31. The reduced overhead text messaging system according to claim 30 wherein the table of capitalization control tokens comprises at least 3 elements.

32. The reduced overhead text messaging system according to claim 31 wherein the table of capitalization control tokens comprises 3 tokens pairs of:
<all capitals>; <all lower case>; and <capitalize first letter>.

33. The reduced overhead text messaging system according to claim 29 wherein the capitalization control token comprises 10 bits when the capitalization control token is not embedded in a selected data type switch token.

34. The reduced overhead text messaging system according to claim 33 wherein the table of capitalization control tokens comprises at least 3 elements.

35. The reduced overhead text messaging system according to claim 34 wherein the table of capitalization control tokens comprises 3 tokens pairs of:
<all capitals>; <all lower case>; and <capitalize first letter>.

36. The reduced overhead text messaging system according to claim 4 wherein the fragment and phrase token is a member of a table of fragment and phrase tokens each representing a compressed element.

37. The reduced overhead text messaging system according to claim 36 wherein the fragment and phrase token comprises 10 bits.

38. The reduced overhead text messaging system according to claim 37 wherein the table of fragment and phrase tokens comprises at least 542 elements.

39. The reduced overhead text messaging system according to claim 37 wherein the table of fragment and phrase tokens comprises less than 542 elements.

40. The reduced overhead text messaging system according to claim 37 wherein the table of fragment and phrase tokens comprises more than 542 elements.

41. The reduced overhead text messaging system according to claim 1 further comprising a message processing computer that receives the original message and a destination identifier and generates, in conjunction with the lossless compression engine, a compressed message including a selective call address.

42. The reduced overhead text messaging system according to claim 41 further comprising a messaging unit that receives the compressed message in response to correlating the selective call address with a predetermined selective call address associated with the messaging unit.

43. The reduced overhead text messaging system according to claim 42 wherein the messaging unit decodes and decompresses the compressed message to recover the original message.

44. The reduced overhead text messaging system according to claim 43 wherein the messaging unit decodes and decompresses the compressed message according to a compression version number received in conjunction with the compressed message.

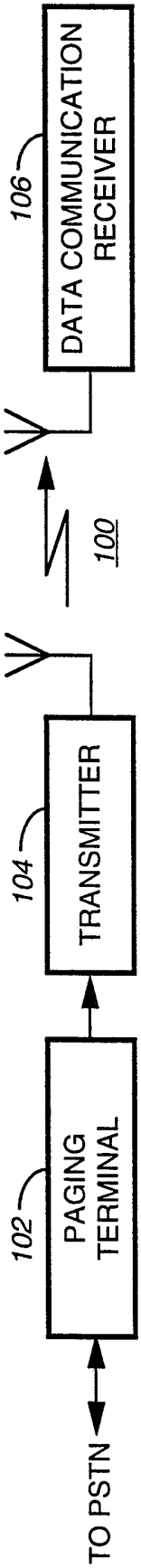

TO PSTN
102
PAGING TERMINAL
104
TRANSMITTER
100
106
DATA COMMUNICATION RECEIVER

FIG. 1

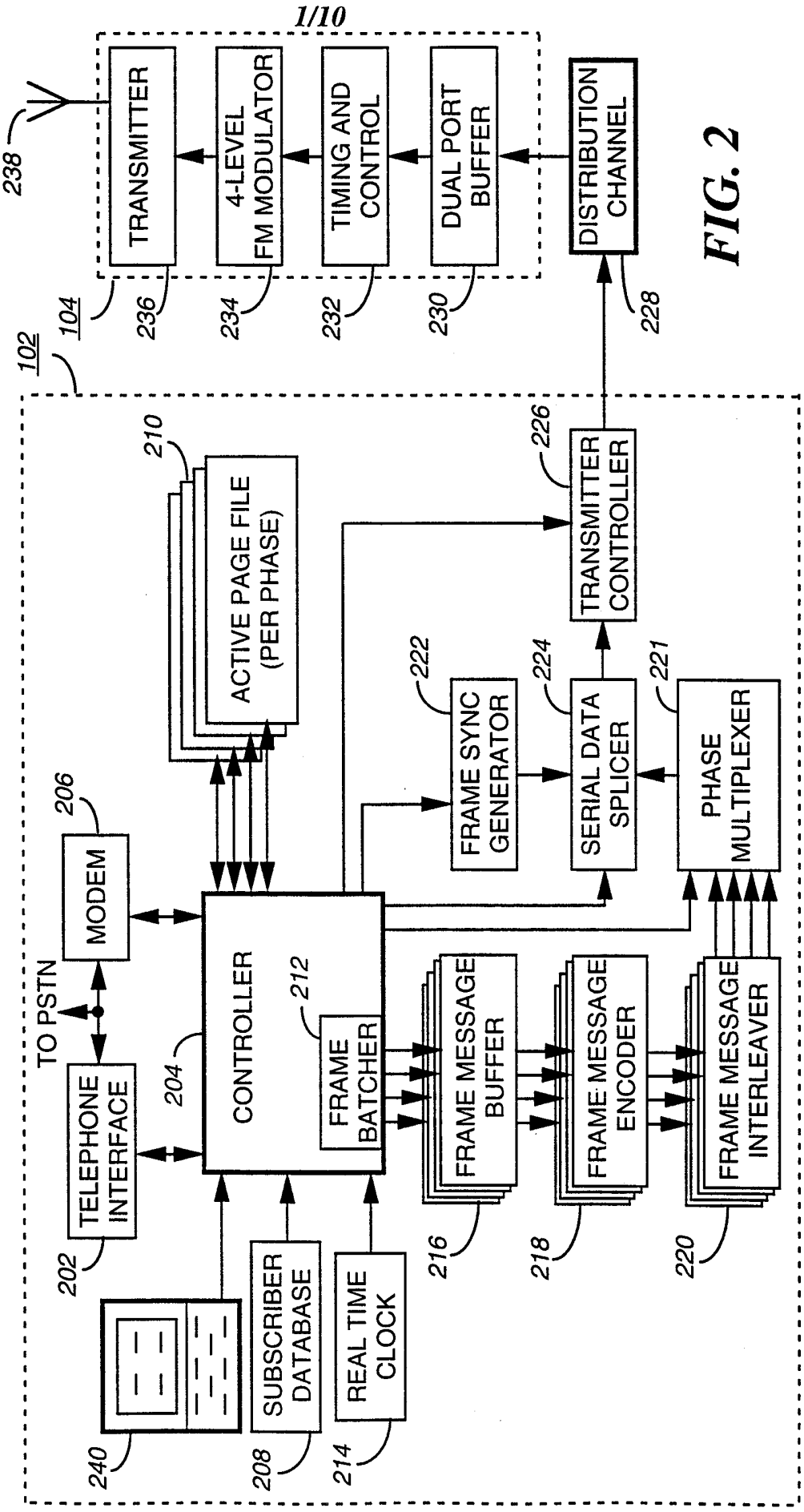

238
TRANSMITTER
4-LEVEL FM MODULATOR
TIMING AND CONTROL
DUAL PORT BUFFER
DISTRIBUTION CHANNEL
236
234
232
230
228
104
102
210
ACTIVE PAGE FILE (PER PHASE)
226
TRANSMITTER CONTROLLER
222
FRAME SYNC GENERATOR
224
SERIAL DATA SPLICER
221
PHASE MULTIPLEXER
206
MODEM
TO PSTN
CONTROLLER
212
FRAME BATCHER
204
FRAME MESSAGE BUFFER
FRAME MESSAGE ENCODER
FRAME MESSAGE INTERLEAVER
TELEPHONE INTERFACE
202
216
218
220
SUBSCRIBER DATABASE
REAL TIME CLOCK
240
208
214

FIG. 2

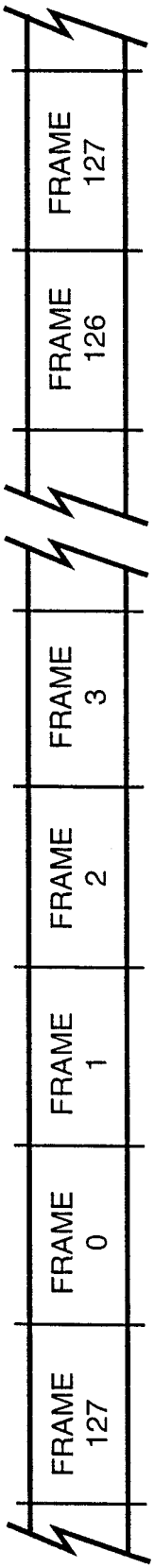


FIG. 3

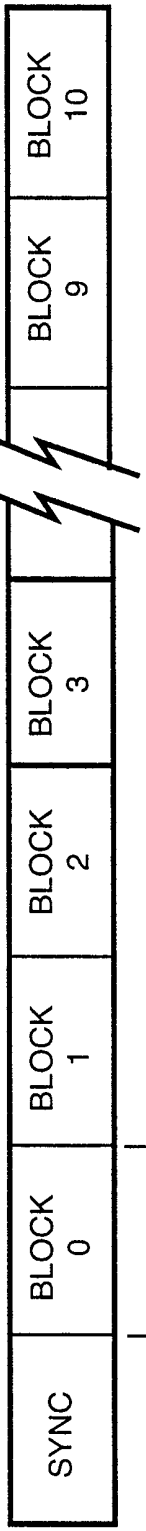


FIG. 4

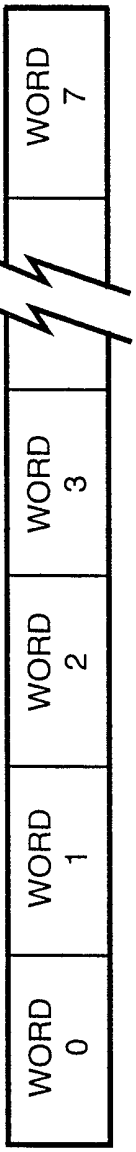


FIG. 5

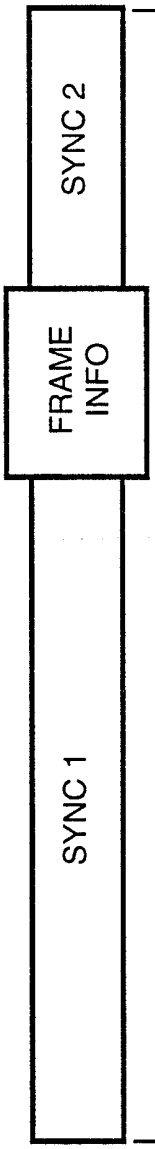


FIG. 6

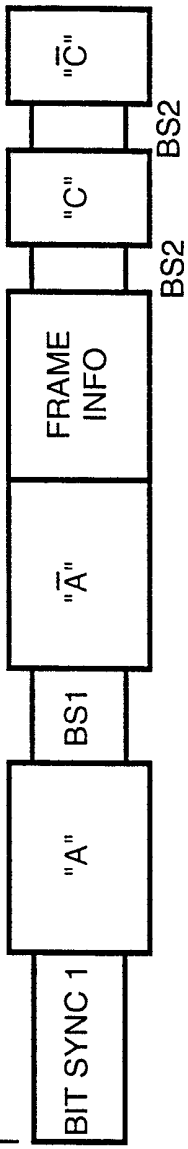
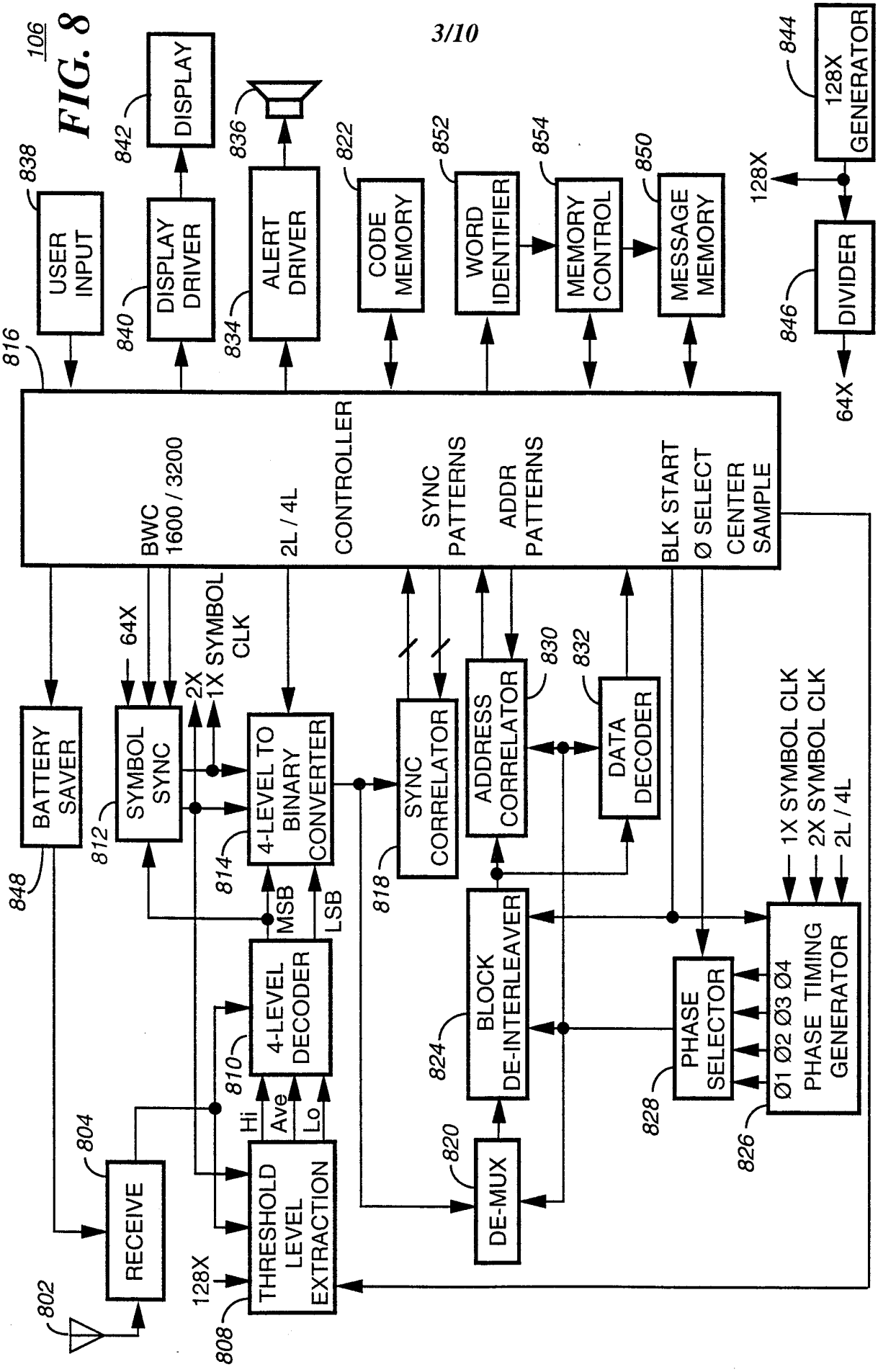


FIG. 7

FIG. 8


106
802
804
806
808
810
812
814
816
818
820
822
824
826
828
830
832
834
836
838
840
842
844
846
848
850
852
854
RECEIVE
THRESHOLD LEVEL EXTRACTION
128X
4-LEVEL DECODER
Hi
Ave
Lo
MSB
LSB
SYMBOL SYNC
64X
2X
1X SYMBOL CLK
4-LEVEL TO BINARY CONVERTER
BATTERY SAVER
SYNC CORRELATOR
ADDRESS CORRELATOR
DATA DECODER
DE-MUX
BLOCK DE-INTERLEAVER
PHASE SELECTOR
Ø1 Ø2 Ø3 Ø4
PHASE TIMING GENERATOR
1X SYMBOL CLK
2X SYMBOL CLK
2L / 4L
CONTROLLER
BWC
1600 / 3200
2L / 4L
SYNC PATTERNS
ADDR PATTERNS
BLK START
Ø SELECT
CENTER SAMPLE
USER INPUT
DISPLAY DRIVER
DISPLAY
ALERT DRIVER
CODE MEMORY
WORD IDENTIFIER
MEMORY CONTROL
MESSAGE MEMORY
128X GENERATOR
DIVIDER
64X

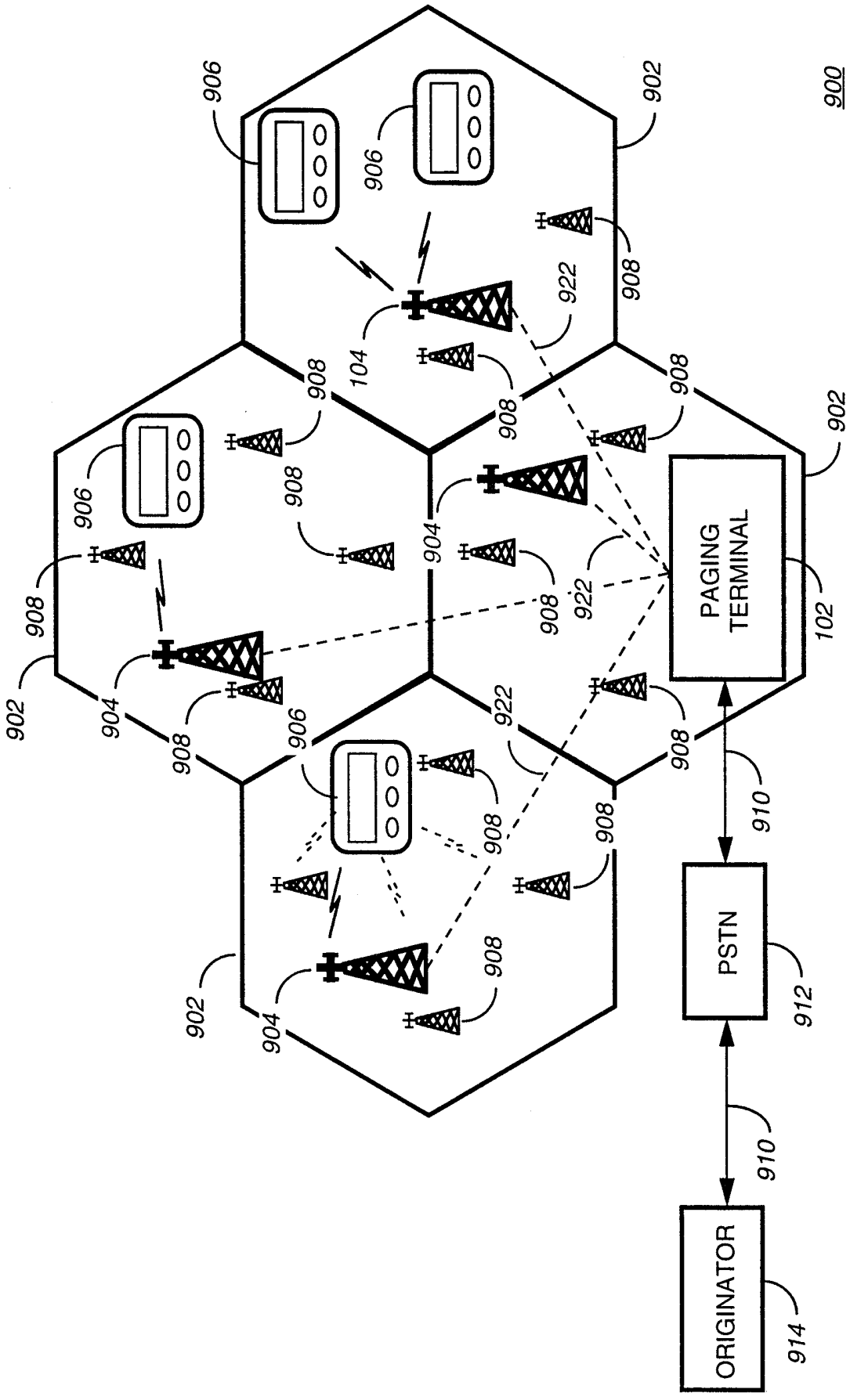

900
902
904
906
908
910
912
914
922
102
104
PAGING TERMINAL
PSTN
ORIGINATOR

FIG. 9

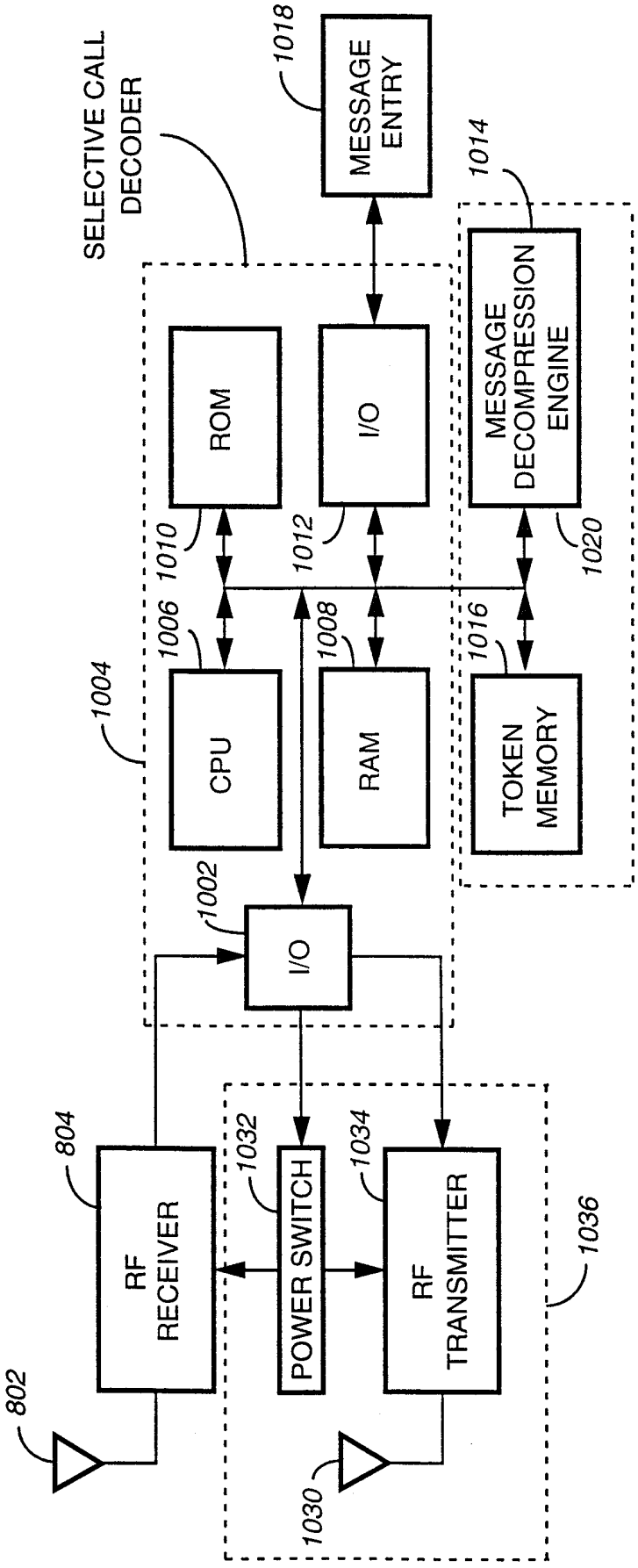

SELECTIVE CALL DECODER
1018
MESSAGE ENTRY
1014
MESSAGE DECOMPRESSION ENGINE
1020
ROM
1010
I/O
1012
1006
1008
1016
CPU
RAM
TOKEN MEMORY
1004
1002
I/O
804
1032
1034
RF RECEIVER
POWER SWITCH
RF TRANSMITTER
1036
802
1030
906

FIG. 10

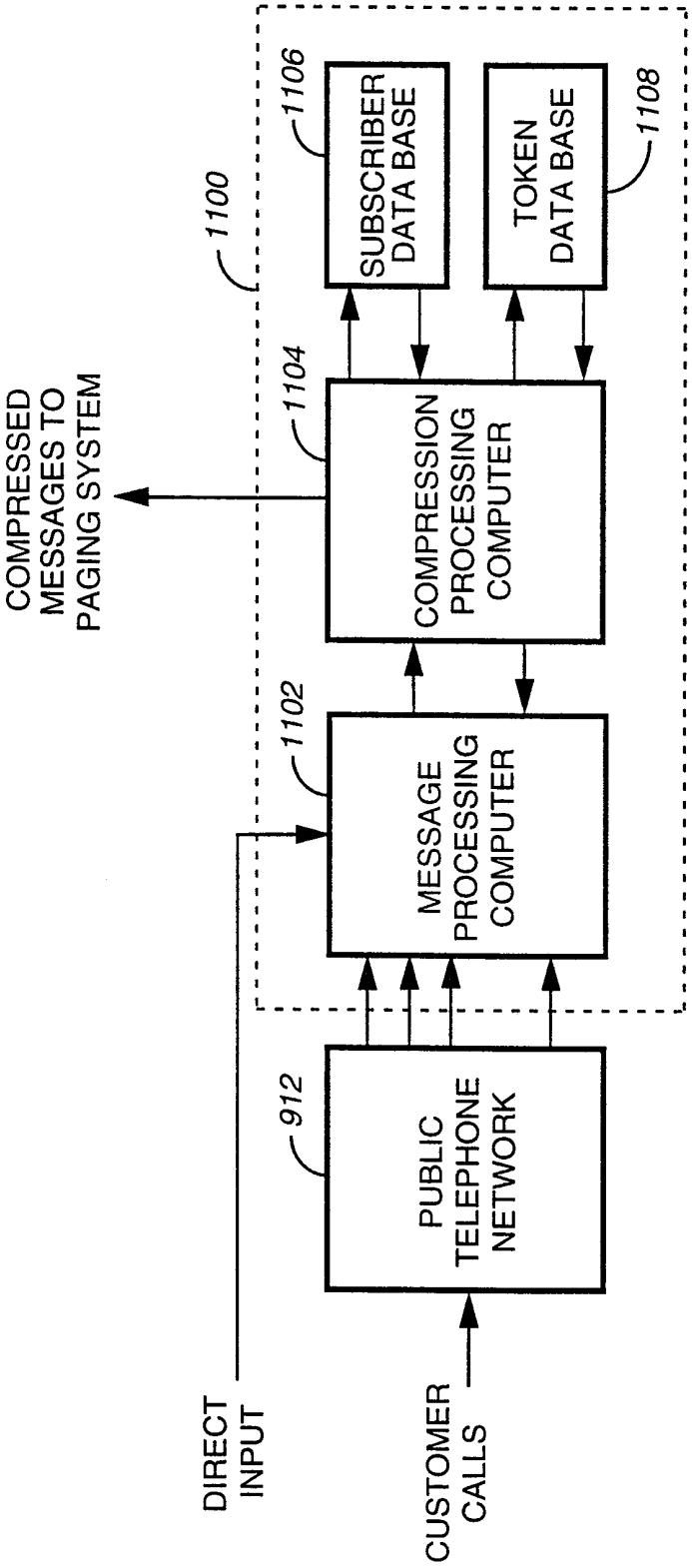

1100
1106
SUBSCRIBER DATA BASE
1108
TOKEN DATA BASE
1104
COMPRESSION PROCESSING COMPUTER
COMPRESSED MESSAGES TO PAGING SYSTEM
1102
MESSAGE PROCESSING COMPUTER
912
PUBLIC TELEPHONE NETWORK
DIRECT INPUT
CUSTOMER CALLS

FIG. 11

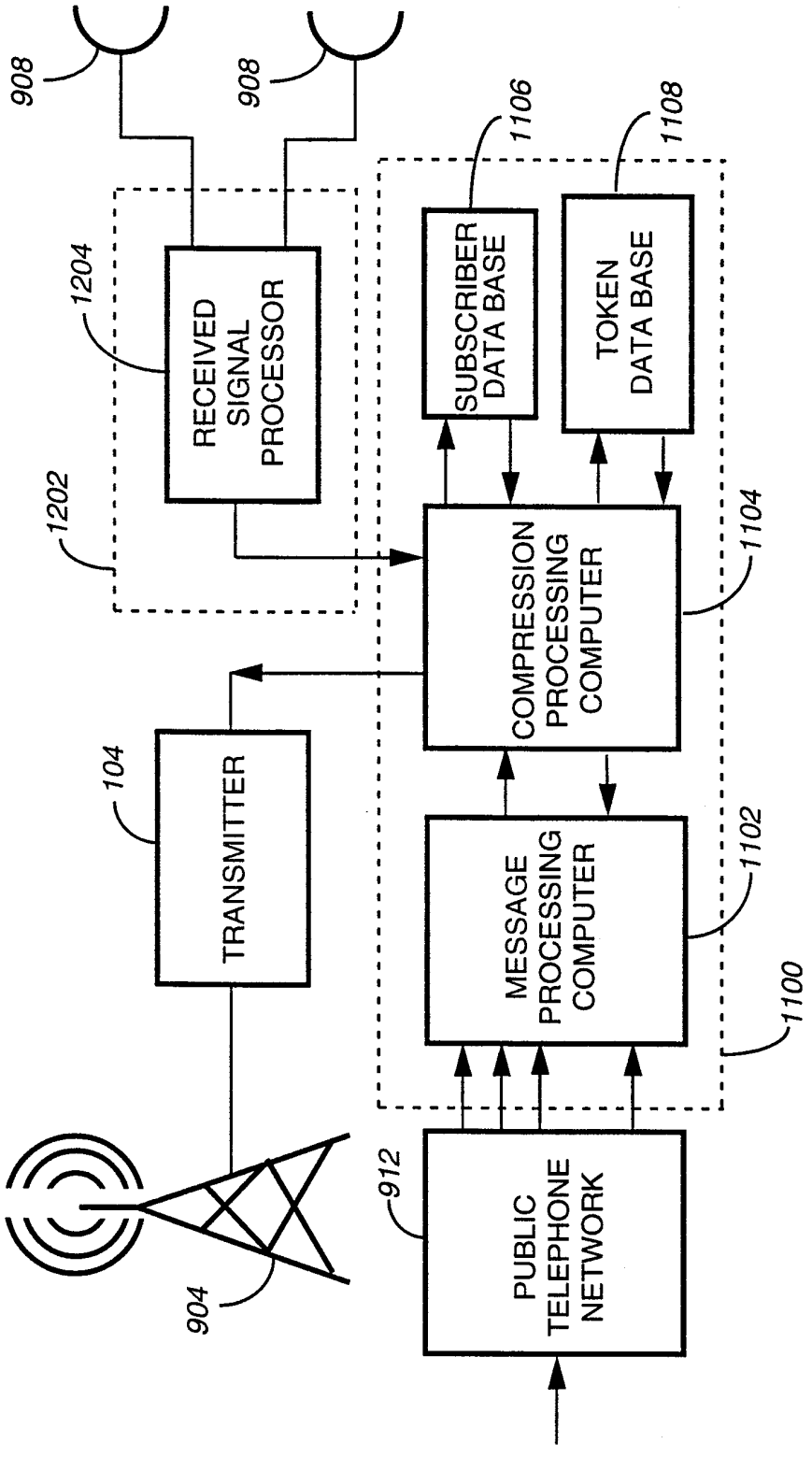

908
908
1106
1108
1204
RECEIVED SIGNAL PROCESSOR
SUBSCRIBER DATA BASE
TOKEN DATA BASE
1202
COMPRESSION PROCESSING COMPUTER
1104
104
TRANSMITTER
MESSAGE PROCESSING COMPUTER
1102
1100
912
PUBLIC TELEPHONE NETWORK
904

FIG. 12

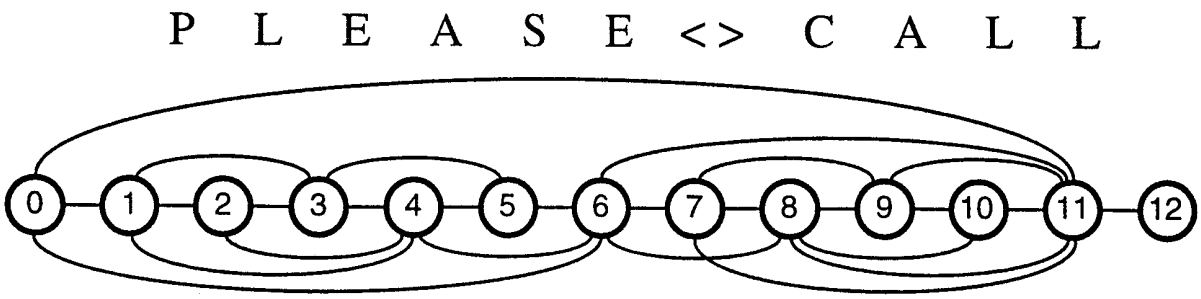
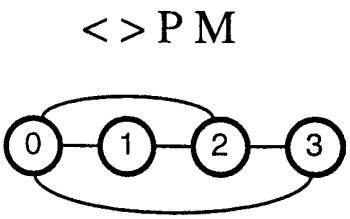


FIG. 13



Stg	P-Stg	Link	S_4	S_5	S_c	n_4	n_c	Σ_4	Σ_5	Σ_T	Σ_c	4	5	T	c
0	-	-	0	0	0	0	0	0	0	0	0	-	0	-	-

FIG. 15

Stg	P-Stg	Link	S_4	S_5	S_c	n_4	n_c	Σ_4	Σ_5	Σ_T	Σ_c	4	5	T	c
0	-	-	0	0	0	0	0	0	0	0	0	-	-	0	-
1	0 5	p	-	5	-	-	-	-	17	-	-	-	17	10	17
	0 6	p	-	-	c	-	1	-	-	-	17	-	-	-	-
	0 T	p	-	-	-	-	-	-	-	10	-	-	-	-	-
2	1 C	I	-	-	c	-	2	-	-	-	24	-	-	20	24
	1 T	I	-	-	-	-	-	-	-	20	-	-	-	-	-
3	2 C	e	-	-	c	-	3	-	-	-	30	-	-	20	30
	2 T	e	-	-	-	-	-	-	-	30	-	-	-	-	-
	1 T	Ie	-	-	-	-	-	-	-	20	-	-	-	-	-
4	3 C	a	-	-	c	-	4	-	-	-	37	-	37	20	37
	3 5	a	-	5	-	-	-	-	37	-	-	-	-	-	-
	3 T	a	-	-	-	-	-	-	-	30	-	-	-	-	-
	2 T	Ie	-	-	-	-	-	-	-	30	-	-	-	-	-
	1 T	Iea	-	-	-	-	-	-	-	20	-	-	-	-	-
5	4 C	s	-	-	c	-	1	-	-	-	37	-	-	30	37
	4 T	s	-	-	-	-	-	-	-	30	-	-	-	-	-
	3 T	as	-	-	-	-	-	-	-	30	-	-	-	-	-
6	5 C	e	-	-	c	-	2	-	-	-	44	-	17	10	44
	5 T	e	-	-	-	-	-	-	-	40	-	-	-	-	-
	4 T	se	-	-	-	-	-	-	-	30	-	-	-	-	-
	0 5	please	-	5	-	-	-	-	17	-	-	-	-	-	-
	0 T	please	-	-	-	-	-	-	-	10	-	-	-	-	-
7	6 C	<sp>	-	-	5c	-	1	-	-	-	25	25	-	20	25
	6 4	<sp>	54	-	-	1	-	25	-	-	-	-	-	-	-
	6 T	<sp>	-	-	-	-	-	-	-	20	-	-	-	-	-
8	7 C	c	-	-	5c	-	2	-	-	-	32	-	-	20	32
	7 4	c	-	-	-	-	-	-	-	30	-	-	-	-	-
	6 T	<sp>c	-	-	-	-	-	-	-	20	-	-	-	-	-
9	8 C	a	-	-	c	-	1	-	-	-	37	-	37	20	37
	8 T	a	-	-	-	-	-	-	-	30	-	-	-	-	-
	8 5	a	-	5	-	-	-	-	37	-	-	-	-	-	-
	7 T	ca	-	-	-	-	-	-	-	30	-	-	-	-	-
	6 T	<sp>ca	-	-	-	-	-	-	-	20	-	-	-	-	-
10	9 C	I	-	-	c	-	1	-	-	-	37	-	-	30	37
	9 T	I	-	-	-	-	-	-	-	30	-	-	-	-	-
	8 T	al	-	-	-	-	-	-	-	30	-	-	-	-	-
11	10 C	I	-	-	c	-	2	-	-	-	44	-	17	20	44
	10 T	I	-	-	-	-	-	-	-	40	-	-	-	-	-
	9 T	II	-	-	-	-	-	-	-	30	-	-	-	-	-
	8 T	all	-	-	-	-	-	-	-	30	-	-	-	-	-
	7 5	call	-	545	-	-	-	-	32	-	-	-	-	-	-
	7 T	call	-	-	-	-	-	-	-	30	-	-	-	-	-
	6 5	<sp>call	-	5	-	-	-	-	24	-	-	-	-	-	-
	6 T	<sp>call	-	-	-	-	-	-	-	20	-	-	-	-	-
	0 5	please call	-	5	-	-	-	-	17	-	-	-	-	-	-
12	1 1	-	-	-	-	-	-	-	-	27	-	-	-	27	-

FIG. 14

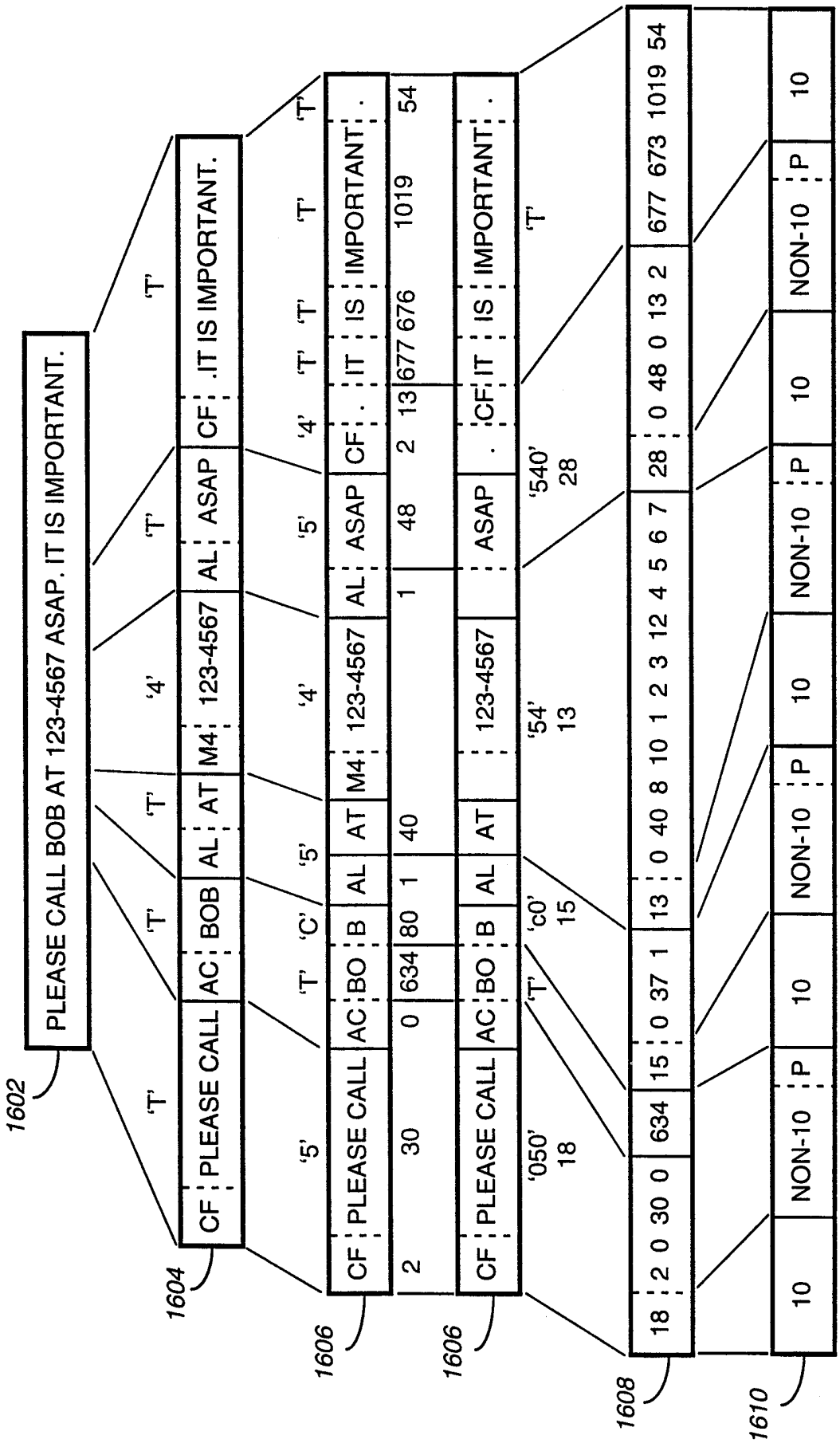

PLEASE CALL BOB AT 123-4567 ASAP. IT IS IMPORTANT.
1602
1604
1606
1606
1608
1610
P: POSSIBLE PADDING LOCATIONS.

FIG. 16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/13293

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 5/00
US CL : 707/527

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)

U.S. : 707/527 341/51

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 practicable, search terms used)

APS SEARCH

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,A	US 5,561,421 A (SMITH et al.) 01 October 1996, cols 4-5, cols 23-24	1-44
Y,P	US 5,838,963 A (GRIFFITHS) 17 November 1998, col 2, lines 35-67	1-44
Y,A	US 5,585,793 A (ANTOSHENKOV et al) 17 December 1996, figure 6, col 2, lines 60-67, col 3, lines 1-20	4-28
Y,A	US 5,325,091 A (KAPLAN et al.) 28 June 1994, col 5, lines 1-40, lines 62-67, col 6, lines 1-35, figures 4-7	4-20
Y,A	US 4,597,057 A (SNOW) 24 June 1986, figures 2, 5	6-10

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"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
"A"	document defining the general state of the art which is not considered to be of particular relevance	"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
"E"	earlier document published on or after the international filing date	"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
"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)	"&"	document member of the same patent family
"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		

Date of the actual completion of the international search 17 SEPTEMBER 1999	Date of mailing of the international search report 25 OCT 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer STEPHEN HONG Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet)(July 1992) ★