



7.)

DOT MATRIX PRINTER AND METHOD FOR STORING AND RETRIEVING BIT MAP DATA

The present invention relates to a dot matrix printer and a method for storing and retrieving bit map data.

To print patterns other than the graphics and characters previously stored as character sets in a ROM of a printer, the user must prepare patterns bit map data corresponding to the characters to be printed, and store the prepared patterns as supplemental character data in a memory of the printer.

Usually, a part of a buffer memory, for example, in the printer, is used for storing the supplemental character data. In this case, it is necessary to provide in advance a memory having a capacity sufficient for storing the data of the largest size expected.

The amount of supplemental character data differs among users. Some users require only a small amount of supplemental character data, while others need a large amount of such data. In the former case, a large part of the memory allocated for storing the supplemental character data is not used and is wasted.

A technique for reducing the memory capacity of the memory used for storing the data by compressing the bit map data of the supplemental character data has been used for factory-installed character patterns. However, it is difficult for general users to employ such a method for storing supplemental character data since special knowledge is required for carrying out such a method.

The present invention has been made in view of the above circumstances, and has as an object the provision of a dot matrix printer which can store supplemental character data prepared by a

REPLACED SHEETS SG

☒	Description	: paged	1	to paged	16
☒	Claim(s)	: paged	1	to paged	4
☒	Drawing(s)	: paged		to paged	
☒	Abstract	: paged		to paged	



user without requiring any complicated steps to be carried out by the user. This object is solved by the dot matrix printer of independent claim 1 and the method of independent claim 7. Further advantageous features, aspects and details of the invention are evident from the dependent claims, the description and the drawings. The claims are intended to be understood as a first non-limiting approach of defining the invention in general terms.

The invention allows that the user can store supplemental character data prepared by the user after the data has been converted into compressed data without requiring any compressing operation to be performed by the user, and which can effectively utilize the available memory.

The invention provides a dot matrix printer which stores bit map data composed of characters and graphics provided by a user, and which reads out the bit map data with code signals received from a host computer. More particularly, the invention provides a dot matrix printer employing an improved technique for storing and reading out the bit map data.

The dot matrix printer of the invention has reduced bit map data storage.

To achieve the above and other aspects of the invention, according to a specific aspect of the invention, a dot matrix printer comprises means for compressing bit map data using a plural number of data compressing methods, means for selecting the data compressed by the compressing method providing the smallest amount of compressed data from among the plural items of data compressed by the plural number of compressing methods and which are output from the compressing means, means for storing



the compressed data selected by the selecting means, and means for restoring the compressed data read out of the storing means to bit map data.

That is, in accordance with a specific aspect of the invention, the bit map
5 data is compressed using a plural number of compressing methods. The amounts of the compressed data resulting from the application of these compressing methods are compared. The compressed data having the smallest size is selected, and then stored in to the storing means. Accordingly, the supplemental
10 character data prepared by the user can automatically be stored in the storing means in the form of compressed data having the smallest size. Accordingly, the memory of the printer can be more effectively utilized.

15 The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate preferred embodiments of the invention, and, together with the following description, serve to explain the objects, advantages and principles of the invention.

20 Fig. 1 is a block diagram showing the overall system of a dot matrix printer constructed according to a preferred embodiment of the present invention;

Fig. 2 is a block diagram showing the functions of the control means used in the printer of Fig. 1, which can be incorporated by means of a microcomputer.



Fig. 3 is a diagram showing the structure of a memory area assigned to one supplemental character in a RAM;

Fig. 4 is a diagram showing an example of bit map data, which is used for explaining the operation of the printer;

5 Fig. 5 is a flow chart showing a sequence of procedural steps for a pre-compression process employed in the printer;

Fig. 6 is a diagram showing the bit map data after pre-compression;

10 Fig. 7 is a flow chart showing a sequence of procedural steps of the data compression process carried out in a first compression mode in the printer;

Fig. 8 is a diagram showing the state of the data compression achieved by the first compression mode;

15 Fig. 9 is a flow chart showing a sequence of procedural steps of the data compression process carried out in a second compression mode;

Fig. 10 is a diagram showing the state of the data compression achieved by the second compression mode;

20 Fig. 11 is a flow chart showing a sequence of procedural steps of the data compression process carried out in a third compression mode; and

Fig. 12 is a diagram showing the state of the data compression achieved by the third compression mode.

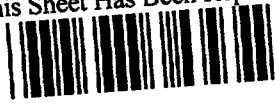


Fig. 1 is a block diagram showing the overall system of a dot matrix printer constructed according to a preferred embodiment of the present invention.

5 In the figure, reference numeral 1 designates a control unit composed mainly of a microcomputer, which receives signals, such as supplemental character data and control commands, from a host computer by way of an interface 2. The control unit 1 is connected to a memory 4 for storing control
10 programs to control the printer functions and a memory device 5 including a ROM for storing bit map data of characters and patterns as defaults and a work RAM. The control unit 1 is
further connected to a driver 7 for transferring bit map data through another interface 6 to a print head 8.

15 Fig. 2 is a block diagram showing the functions of the control unit 1 (microcomputer) shown in Fig. 1. A data analyzer section 10 analyzes signals received from the host computer to produce supplemental character data, address signals for the memory, memory select signals and control
20 signals.

The address signal output from the data analyzer section 10 is applied to a RAM 17 and a ROM 16 in which fonts as defaults are previously stored for addressing them. The select signal and control signal from the data analyzer section
25 10 are applied to the ROM 16 and the RAM 17 for selecting the RAM 17 or the ROM 16 and for controlling the read/write



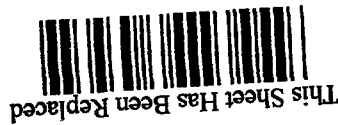
operation of the selected one. The supplemental character data from the data analyzer section 10 is input to a bit map data generator section 11. The bit map data generator section 11, which contains a buffer memory, generates bit map data of the supplemental character data, and assembles a map of the bit map data in an X-Y coordinate area of the buffer memory. The bit map data mapped in the buffer memory of the bit map data generator section 11 is read therefrom by a null-series removal section 12. The null-series removal section 12 checks whether or not, of the bit map data (of Fig. 4, or example), grouped data (referred to as data series) arrayed serially and in the direction (vertical direction in Fig. 4) of arraying dot forming elements (e.g., print pins) of the print had 8 are all the data (referred to as null data) not forming dots (viz., which is null). If data series (referred to as null data) consisting of only null data is present, the null-series removal section 12 removes the data (this removal process is called a pre-compression process).

The pre-compressed bit map data output from the null-series removal section 12 is input to a data compressor 13 operable in plural types of data compression modes. The data compressor 13 compresses the pre-compressed bit map data in the respective compression modes. The bit map data (compressed data) compressed in the compression modes, together with the pre-compressed bit map data not yet compressed (referred to as uncompressed data), is transferred to a comparator 14. The



comparator compares the amounts of the compressed data and the uncompressed data, and selects the compressed data or uncompressed data having the smallest size. The selected data is transferred to a register section 15. Then, the data,
5 together with information data attached thereto, such as half information, left-space information, right-space information, and compression mode information (as will be discussed in detail later) is stored into the RAM 17.

The data read out of the ROM 16 or the RAM 17 is
10 transferred to a compression mode discriminator 18. The compression mode discriminator 18 discriminates the data compression mode by which the compressed data was compressed, determines whether or not the data is compressed, and transfers the read-out data to one of restoring modules 19a to 19d of a
15 compressed-data restoring section 19 in accordance with the result of the compression-mode discrimination. The compressed-data restoring section 19 restores the compressed data from the RAM 17 to the pre-compressed bit map data using the attendant data of the half information, left-space information, right-space information, and compression mode information. The bit
20 map data from the compressed-data restoring section 19 is input to a null-series restoring section 20, which in turn adds the null series, which was removed by the null-series removal section 12, to the bit map data. As a result, the compressed
25 data is restored to the original bit map data. The original



bit map data is transferred through the interface 6 to the print head 8 where it is printed out.

In the dot matrix printer thus constructed, when receiving a supplemental-character-data request from the host computer, the data analyzer section 10 acknowledges the request and then enables the RAM 17 by a select signal. The supplemental character data from the host computer is compressed by the best data compression mode as stated above, and transferred as compressed data to the RAM 17. At the same time, the RAM 17 is addressed by an address signal. The data is written into the memory location of the specified address in the RAM under the control signal. In this way, the compressed supplemental character data attendant with the information data is registered in the RAM 17. Details of the processing of the supplemental character data will be described hereinafter.

The bit map data generator section 11 converts the supplemental character data received from the host computer into bit map data representative of a dot pattern. Subsequently, the bit map data is transferred to the null-series removal section 12 where the null series is removed from the data (i.e., the data is pre-compressed).

The pre-compression is carried out as follows. The number of successive null series is counted from the left end of the bit map data. The counted number of the null series is treated as left-side null data. The null series present successively from the left end of the bit map data are removed



in the counted number. Then, as for half information, the null-series removal section checks whether or not the null series are present every second data series in the remaining bit map data. If the null series are absent, the null-series removal section turns off the half-dot present flag. If those
5 are present every second data series, the null-series removal section turns on the half-dot present flag, and counts the number of successive null series from the right end of the bit map data treating the counted number of the null series as
10 right null information, and removes the counted number of the null series present successively from the right end of the bit map data. When the half-dot present flag is in an ON state, for instance, the null series are removed. When the half-dot present flag is in an OFF state, no further removal of null
15 series is carried out. After the completion of the pre-compression process, the number of bytes of bit map data is counted.

Subsequently, the pre-compressed bit map data is transferred to the data compressor 13 where it is further
20 compressed. In this embodiment, three data compression modes are employed, which operate in the following ways.

A first data compressor (first compression mode) 13a checks whether or not the pattern of the whole pre-compressed bit map data is symmetrical with respect to the vertical center
25 line. If it is symmetrical, the data is divided, with respect



to the center line, into two portions, right and left data. Only one of the right and left data is employed.

5 A second data compressor (second compression mode) 13b calculates the exclusive-OR logical sum of the right and left data bytes adjacent to each other (the data byte means the data group forming one byte in the data series; in the case of Fig. 4, the upper half and the lower half of the data series are respectively the data bytes) to check whether or not the data bytes are equal to each other. If the adjacent data bytes are the same, one (e.g., the right data) of the data bytes is removed. If the two byte data are different, both byte data are left.

... A third data compressor (third compression mode) 13c checks whether data byte (referred to as null byte) consisting of only null data is present or absent. In this compression mode, the symmetry and the equality of the adjacent data bytes are not checked. If the null byte is present, the third compressor removes the null byte and leaves the remaining bytes.

20 The sizes (data amounts) of the data thus compressed in the respective compression modes are calculated. The sizes of the three compressed data, as well as the size of the uncompressed data, are compared by the comparator 14. Of these compression modes, the compression mode providing the smallest data size is determined to be the best data compression mode. 25 The compressed data having undergone the best compression mode,



together with the attendant data, such as half information, left-null information, right-null information, and compression mode information, is transferred to the register section 15 and then stored into the RAM 17.

5 Fig. 3 is a diagram showing the structure of a memory area assigned to one supplemental character in a RAM. As shown, half information data is stored in a subaddress 0, left-null information data in the subaddress 1, right-null information data in the subaddress 2, and compression mode information data in the subaddress 3. Removal byte information is stored in the subaddresses 4 to m . The removal byte information indicates the addresses of the data byte removed by the best compression mode. The bits in this memory area correspond to the byte addresses in the bit map data. Data "1" is stored in the bits of the addresses of the removed byte data. The remaining memory area from the subaddresses ($m + 1$) to n store the compressed data.

10 To print the supplemental characters registered in such an order, the data analyzer section 10, which receives a supplemental-character print request from the host computer, sends select, address, and control signals to the RAM 17. The compressed data of the supplemental characters, and the attached data of half information, left-null information, right-null information, and compression mode information, are read out of the RAM 17, and transferred to the compression mode discriminator 18. Upon receipt of the information data, the



compression mode discriminator 18 transfers the supplemental character data to the restoring module which corresponds to the best compression mode. Specifically, when the data is compressed by the first compression mode, the compression mode discriminator 18 transfers the data to the first restoring module 19a. When the data is compressed by the second compression mode, the data is transferred to the second restoring module 19b, and so on. The data, after being restored by the compressed-data restoring section 19, is transferred to the null-series restoring section 20. In the null-series restoring section 20, the null series are restored on the basis of the attendant data of half information, left-null information, right-null information, and compression mode information. The original bit map data thus restored is sent to the printer for printing out the bit map data.

A specific operation of the dot matrix printer as stated above will be described using an example of the bit map data shown in Fig 4.

An operation flow of the process (pre-compression process) of removing the null series, which follows the compression process, is shown in Fig. 5. The null-series removal section checks the number of right null series. In this instance, data series A and B are the left null series. Accordingly, 2 as the number of null series is registered as the left null information, and these null series are removed. Then, the null-series removal section checks the remaining bit



map data successively from the right side to the left side to determine whether or not the null series is present every second data series. In this instance, the null series "d", "b", ..., "F", and "D" are present every second data series, so
5 that the half-dot present flag is turned on. Then, the null series "e" and "d" are detected, and these two null series are registered as right null information and removed. Since the half-dot present flag is in an ON state, the null series are removed every other data series. In this way all the null
10 series are removed, so that bit map data as shown in Fig. 6 is obtained. In this bit map data, half of one data series corresponds to one byte. Accordingly, the data amount of the bit map data is 28 bytes. Even-numbered byte addresses are assigned to the upper data byte, and odd-numbered byte
15 addresses are assigned to the lower data byte.

Subsequently, the data of Fig. 6 is compressed by the data compressor 13. Firstly, the first compression mode will be described. The operational flow of the first compression mode is shown in Fig. 7. The state of data compression is
20 shown in Fig. 8. The first compressor checks the overall pattern of the bit map data to determine whether or not the pattern is symmetrical with respect to the center line. If it is symmetrical, the bit map data is divided into right and left data, and a symmetrical flag indicating the symmetry of the
25 pattern is set. One (e.g., left data) of the right and left data is selected, and the amount of data is calculated. If it



is asymmetrical, this compression mode does not compress the data. A shaded portion in Fig. 8 indicates the data byte to be removed by the compression process. In this instance, since the amount of the data before compression is 28 bytes, the amount of data after compression is half of the former, i.e., 14 bytes.

The second compression mode will be described. The operational flow of the second compression mode is illustrated in Fig. 9. The state of data compression by the second compression mode is shown in Fig. 10. The second data compressor compares the right and left data bytes to check whether or not those data bytes are equal to each other. If those data bytes are equal, one of the data bytes is removed. If those data bytes are not equal, the second compressor advances to the next data series and compares new adjacent data bytes. A comparison is made of all pairs of adjacent data bytes. In this example, when checking the data from the left side, the bytes at the addresses 0, 2, 4, 6, and 8 are not equal, and hence are left as they are. The data bytes at the addresses 8 and 10 are equal, so that the data byte at the address 10 is removed. The data bytes at the addresses 8 and 12 are not equal, and thus are left as they are. Then, the data bytes at the addresses 12 and 14 are compared. These bytes are equal, so that the bytes at the address 14 is removed. The comparing process progresses until finally the bytes hatched in Fig. 10 are removed. In this instance, the



amount of the original data is reduced by 6 bytes, so that the amount of the compressed data is 22 bytes.

5 The third compression mode will be described. As seen from the flow chart of Fig. 11, when the bits of the data byte are all spaces, this compression mode removes the space bytes. The removal process is applied to all data bytes. In this instance, the bytes of which the bits are all space are only the bytes at the addresses 7 and 21, which are thus removed. Since the amount of the data before compression is thus reduced
10 by two types, the amount of the data after compression is 26 bytes.

When comparing the amounts of the data after
compression by the first to third compression modes, it is seen that the data amount achieved by the first compression mode is
15 the smallest. Accordingly, the data compressed by the first compression mode is registered in the RAM 17.

In the above-mentioned embodiment, the best compressing method is selected from among the three compressing methods. It is evident though that other compressing methods may be used
20 for the present invention. For example, the patterns of the data bytes separated by one byte from each other may be compared for checking whether or not these patterns are equal. Where three or more of the same pattern successively recur, the first byte and the number of bytes having the same patterns as
25 that of the first byte may be registered as the compressed data.

As seen from the foregoing description, in the present invention, the supplemental character data is automatically compressed and registered in the memory, which is similar to the normal procedure for preparing supplemental characters. Therefore, the memory capacity can be effectively used.

The foregoing description of a preferred embodiment of the invention has been presented for purposes of illustration and description.



CLAIMS

1. A dot matrix printer comprising:

a plurality of data compressors (13a, 13b, 13c..) for compressing bit map data each by a different data compressing method;

a data selector (14) for selecting the data compressed by the one of said data compressors (13a, 13b, 13c..) which provides the smallest amount of compressed data from among plural items of data compressed by said plurality of data compressors and output from said data compressors (13a, 13b, 13c..);

a memory (17) for storing the compressed data selected by said data selector (14); and

a data restorer (19a,20) for restoring the compressed data, read out from said memory (17) to bit map data.

2. The dot matrix printer according to claim 1, further comprising a bit map data generator (11) for generating bit map data using pattern data of character/graphic patterns received from an external device, and applying the generated bit map data to said data compressors (13a, 13b, 13c..).

3. The dot matrix printer according to claim 1 or 2, further comprising a pre-compressor (12) for removing null data satisfying predetermined conditions from original bit map data, bit map data output by said pre-compressor (12) being applied to said plurality of data compressors (13a, 13b, 13c) for compression thereby.

4. The dot matrix printer according to anyone of the preceding claims, wherein said memory (17) stores compressed data



together with compressing-method information indicative of the selected data compressor (13a, 13b, 13c..) and removed-data information indicative of positions of data deleted by said selected data compressor.

5. The dot matrix printer according to claim 4, in which said data restorer (19a, 20) employs a plural number of data restoring modules (19a, 19b, 19c...) corresponding to the plural number of said data compressors (13a, 13b, 13c..), said data restorer selecting a one of said restoring modules (19a, 19b, 19c..) corresponding to the selected data compressor (13a, 13b, 13c) on the basis of compressing method information read out of said storing means (17).
6. The dot matrix printer according to claim 5, wherein said data restorer (19a, 20) restores deleted data on the basis of deleted data information read out of said storing means (17).
7. A method for storing and retrieving bit map data in a dot matrix printer especially according to any one of the preceding claims, comprising the steps of:

compressing bit map data by a plural number of different data compressing methods; selecting the data compressed by the compressing method which provides the smallest amount of compressed data; storing the selected compressed data; and

reading out and restoring compressed data read out of said storing means to bit map data.

8. The method according to claim 7, further comprising, prior to said step of compressing, the step of generating bit map data using pattern data of character/graphic patterns received from an external device.



9. The method according to claim 7 or 8, further comprising, prior to said step of compressing, pre-compressing said bit map data by removing null data satisfying predetermined conditions.
10. The method according to any one of claims 7 to 9, wherein said step of storing further comprises storing said compressed data together with compressing-method information indicative of the selected compressing method and removed-data information indicative of positions of data deleted in said step of compressing.
11. The method according to any one of claims 7 to 10; wherein, in said step of compressing, one of said said plural number of data compressing methods comprises the steps of:

checking whether or not a pattern of whole precompressed bit map data is symmetrical with respect to a vertical center line thereof;

if said pattern is symmetrical, dividing said precompressed bit map data with respect to said center line into right and left data; and providing only one of said right and left data for said storing step.
12. The method according to any one of claims 7 to 11, wherein, in said step of compressing, one of said said plural number of data compressing methods comprises the steps of: calculating an exclusive-OR logical sum of right and left data bytes adjacent to each other in the pre-compressed bit map data to check whether or not the data bytes are equal to each other; if the adjacent data bytes are the same, removing one of the adjacent data bytes prior to said storing step; and if the adjacent data bytes are different, providing both adjacent data bytes for said storing step.



13. The method according to any one of claims 7 to 12, wherein, in said step of compressing, one of said said plural number of data compressing methods comprises the steps of: checking whether data byte consisting of only null data is present or absent; and if byte data consisting of only null data is present, removing the byte data consisting of only null data prior to said storing step.
14. The method according to any one of claims 7 to 13, further comprising, prior to said step of compressing, pre-compressing said bit map data by the steps comprising: counting a number of successive null series from the left end of the bit map data and treating the counted number as left-side null data; removing null series present successively from said left end of said bit map data; checking whether or not null series are present every second data series in remaining bit map data; if null series are absent in said remaining bit map data, setting a half-dot present flag in an OFF state, and if null series are present every second data series, setting said half-dot present flag in an ON state; counting a number of successive null series from the right end of said bit map data and treating the counted number as right null information; removing the counted number of null series present from said right end of said bit map data successively from said right end of said bit map data; when said half-dot present flag is in said ON state, removing said null series.