

PROCESSING DATA

This invention relates to a data processing apparatus and a method used therein for correcting data errors.

In a variety of data processing apparatus, such as personal computers or word processors, the disk-shaped recording medium, such as the floppy disk, is widely used as the external storage medium.

The data writing and reading in the disk apparatus is usually so performed that the host computer issues a series of control instructions to the disk for sequentially actuating a variety of functional blocks such as rotary drive means or data processing means and the computer operation is performed sequentially in such a manner that, after the end of a predetermined control operation is ascertained, the computer proceeds to the next control operation.

In the conventional floppy disk apparatus, it is therefore up to the host computer to control the operation of the disk apparatus ~~since~~ from the start until the end of data recording and reproduction. Thus a major portion of the processing capacity of the host computer is dedicated to controlling the disk operation such that the processing efficiency of the host computer is lowered considerably.

On the other hand, when correcting the errors of the data recorded with error correction codes, flags are employed in the error correction or parity codes in the conventional

triple erasure correcting routine in such a manner that, although the state of error correction is generally satisfactory for random errors, the risk is high that a double error correction routine is executed on occurrence of burst errors, such that, when a 3-symbol error, for example, is taken for a 2-symbol error, the possibility is high that the correction is not feasible and the errors remain uncorrected.

It is a general object of the present invention to provide a data processing apparatus free of the above problems and a method employed therein for correcting data errors.

It is another specific object of the present invention to provide an error correction method wherein the error correction capability for the burst errors may be improved.

According to the present invention there is provided an apparatus for receiving digital data encoded in an error correction code and performing an error correction decoding, comprising:

- 5 a first memory area for storing received data,
 a second memory area for storing readout flag data during writing of said received data into said first memory area,
 processing means for performing said error correction
10 decoding on said received data, and
 a third memory area for storing flag data corresponding to a result of said error correction decoding, wherein
 said second memory area and said third memory area
15 are provided in the same byte within a single memory,
 said second memory area and said third memory area are set from a first value to a second value during writing of said received data in said first memory area, and
 said second memory area is set to said first value
20 during said error correction decoding of said received data while said third memory area is set to a value corresponding to the result of said error correction decoding.

The present invention will be further described hereinafter with reference to the following description of an
25 exemplary embodiment and the accompanying drawings, in which:

Figs. 1 to 4 are diagrammatic views showing data formats employed in the disk data recording/reproducing to which the present invention is applied.

Figs. 5A and 5B are diagrammatic views showing sub-
30 code data formats composed of the sub-data having the formats shown in Figs. 3 and 4.

Fig. 6 is a diagrammatic view showing the arrangement of the error correction codes having the format shown in Fig. 4.

Fig. 7 is a block diagram showing the disk data recording/reproducing apparatus to which the present invention is applied.

Fig. 8 is a block diagram showing the memory control unit shown in Fig. 7.

Fig. 9 is a diagrammatic view showing a memory map of the buffer memory shown in Fig. 7.

Figs. 10A and 10B are diagrammatic views for understanding the read correction flag area on the memory map shown in Fig. 9.

Fig. 11 is a flow chart showing an example of application of the error correction algorithm of the present invention to the correction of the coding data.

Before proceeding to description of the disk data recording/reproducing apparatus to which the data processing apparatus and the error correcting method according to the present invention are applied, the data formats employed therein will be explained by referring to Figs. 1 to 6. However, those formats are pertinent directly to the present invention are not explained for simplicity.

First, the data format for the floppy disk will be explained.

Track Format

Fig. 1 shows a track format. Each track is formed by an index and four sectors #0 to #3. Thus, from the start position on, each track is constituted by a gap 0 (pre-index gap), index, gap 1 (post-index gap), sector #0, gap 2 (sector gap), sector #1, gap 2 (sector gap), sector #2, gap 2 (sector gap), sector #3, and a gap 3 (the last gap). The index is a

continuous Tmax signal. It is noted that one frame (F) is composed of 44 bytes (B), in this order^e, with each byte including 10 channel bits.

PG denotes a rotational phase detection signal that is produced upon rotation of a spindle motor adapted for rotationally driving the floppy disk. The signal is correlated with the index.

Sector Format

Fig. 2 shows a sector format. Each sector is formed by a preamble data "2BH", a sync frame, sub-frame, coding data (data frames 0 to 127 through n, and a post-amble data "2BH", in this order, and has a length of 5765 bytes.

Sub-frame Format

Fig. 3 shows a sub-frame format. Each sub-frame is formed by sync data, sub-data, frame address, parity codes for the sub-data and address, mode data, track number, sector number (#0, #1, #2 or #3), head number, copy protection code, reserve area and parity codes (4 symbols) for data following the frame address and mode data, in this order, and has a length of 44 bytes.

Data Frame Format

Fig. 4 shows a data frame format. Each data frame is formed by sync data, sub-data, frame address, parity codes for sub-data and address, coding data (32 symbols), C_2 and C_1 parity codes (each 4 symbols) as later described, in this

order, and has a length of 44 bytes.

Sub-Code Format

The sub-code is formed as a sub-code assembly formed by 128 bytes each being the 1-byte sub-data of the data frame shown in Fig. 4A and each being collected from each of the 128 data frames for one sector, as shown in Fig. 5A. It is constituted by the sub-codes #0, #1, #2 and #3. As shown in Fig. 5B, each sub-code is formed by a mode data, track number, sector number, head number, copy protection code, reserve area and parity codes (4 symbols) for the above data, in this order, and has a length of 32 bytes.

Data Frame Error Correction Code Format

The C_2 and C_1 parity codes of the above described data frame will be explained by referring to Fig. 6, wherein, among the data constituting one sector, the data frames taking part in the formation of the C_2 and C_1 parity codes and the sub-frame taking part in the formation of the C_1 parity code, are shown. The C_2 parity codes are constituted by plural symbols of the C_2 series or the coding data obtained on interleaving from a large number of symbols arranged in a matrix in a direction extending from the upper left towards the lower right in the figure. The C_1 parity codes are constituted by plural symbols of the C_1 series extending vertically from the upper side towards the lower side in the figure, that is, the frame address, coding data

and the C_2 parity codes. For these C_1 and C_2 parity codes, the Reed-Solomon codes, for example, are employed.

Disk Data Recording/Reproducing Apparatus

The apparatus for recording and/or reproducing the data having the above described data formats on or from the disk will be explained by referring to Fig. 7.

The data to be recorded are supplied from a host computer 1 to a buffer memory 2 by way of an interface circuit 11 and a disk controller 12 within a disk control section 10. An S-RAM having the storage capacity of the order of 8 k bytes for each sector, for example, is employed as the buffer memory 2. The data written into the buffer memory 2 are subjected to an encoding processing shown in Fig. 6, that is, the formation and annexation of the C_2 and C_1 parity codes, by an error correction processor 13 having the encoding and decoding functions. The encoded output from the buffer memory 2 is supplied to a modulating circuit 14, where it is subjected to a processing such as 8/10 conversion, before it is supplied to a magnetic head 4 via recording/reproducing circuit 3 so as to be recorded on a floppy disk 5. The floppy disk is driven rotationally at 3600 rpm, for example, by a spindle motor 7 controlled by a microcomputer 6 adapted to control the mechanical system. The movement of the magnetic head 4 is also controlled by the microcomputer 6. A serial/parallel interface 15 is provided

between the microcomputer 6 and the disk controller 12 to effect serial communication. It will be noted that the recording/reproducing circuit 3 also has the function as an erasure circuit.

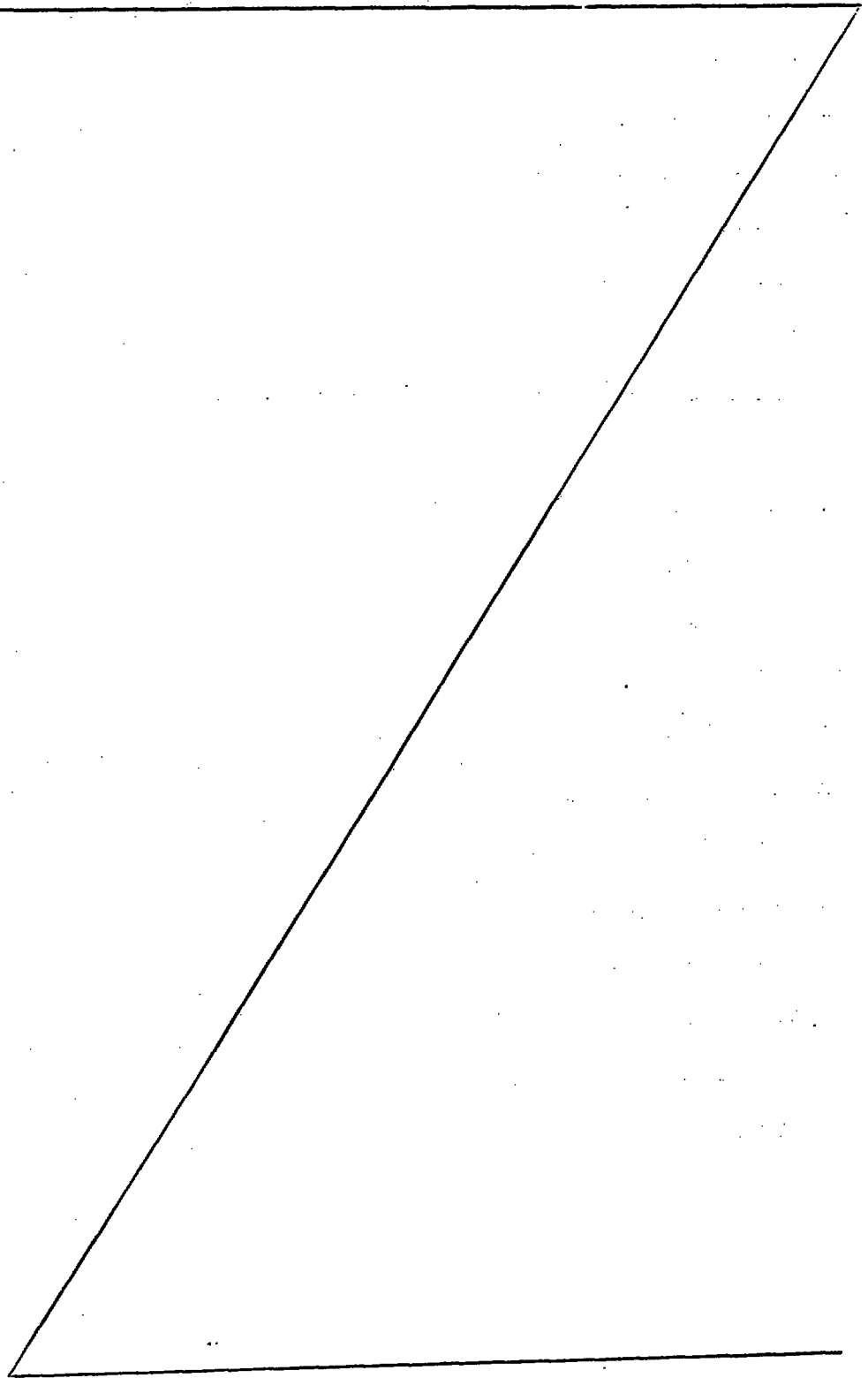
The data read-out by the magnetic head 4 from the floppy disk 5 is supplied via recording/reproducing circuit 3 to a demodulating circuit 16. These data are subjected to a processing including 10/8 conversion in the demodulating circuit 16 after which they are sent to and written in the buffer memory 2. The one-sector data written in the buffer memory 2 are subjected to a predetermined decoding, such as error correction by the C_1 and C_2 parity codes and removal of redundancy bits, before they are supplied via disk controller 12 and interface circuit 11 to the host computer 1.

The disk controller 12 is constituted by a microprogram control section 12a and an instruction command section 12b. The microprogram control section 12a interprets or construes simple instructions supplied from the host computer 1 via interface circuit 11 to supply the micro-codes to the instruction execute section 12b for controlling a series of write/read control operations.

The control data formed in the disk controller 12 is supplied via internal bus to a memory management unit 17.

The disk control section 10, including the interface circuit 11, disk controller 12, an error correction processor

13, demodulating circuit 16, memory management unit 17 and the serial/parallel interface 15, may comprise, for example, a one-chip LSI (large-scale integrated circuit).



Memory Management Unit

The memory management unit 17 responsible for address management of the buffer memory 2 and having the configuration as shown in Fig. 8 is able not only to read or write data on the sector basis in the aforementioned recording or reproducing mode, but also to read or write any desired number of data other than the number of one-sector unit data of 4096 bytes or the number of unit data prescribed by the operating system (OS) of the host computer 1, from and to desired addresses of the buffer memory 2, by way of transferring data between the buffer memory 2 and the host computer 1.

The memory management unit 17 is composed of a start address register 30 for previously storing data indicating the access start address when an access is had to the buffer memory 2 from the host computer 1 to effect data transfer to write or read data, that is, the transfer start address data, a data number register 31 for previously storing the number of the transferred data, a memory address counter 32 for presetting the transfer start address data stored in the start address register 30, a data number counter 33 for presetting data concerning the number of the data stored in the data number register 31, and a control circuit 34 for preset and count control of the memory address counter 32 and the data number counter 33. The unit 17 performs the memory

control of the buffer memory 2 in the following manner.

When accessing the buffer memory 2 from the host computer 1 to effect data transfer to write or read data, the memory management unit 17 is previously supplied with data indicating the transfer start address data and the number of the transferred data from the host computer 1. The unit 17 also has transfer start address data and data number data stored in the start address register 30 and the data number register 31, respectively. Before starting data transfer, the control circuit 34 causes the transfer start address data and the data indicating the transferred data to be preset in the memory address counter 32 and in the data number counter 33 and accesses to the buffer memory from the address indicated in the transfer start address data preset in the memory address counter. The data transfer is then started. Each time one-byte data transfer is completed, the control circuit 34 causes count pulses to be supplied to the memory address counter 32 and the data number counter 33 to increment the memory address counter 32 while simultaneously decrementing the data number counter 33. Thus the data is transferred until the value of the data number counter 33 is reduced to zero, that is, a number of data equal to the value preset in the data number register 31 is transferred, before the data transfer is terminated.

Thus the buffer memory 2 may be accessed from the host

computer 1 from a given address to another given address to write or read a desired number of data, so that, when transferring a number of data other than the number equal to a number raised to the powers of 2 or an integral number of time thereof, any wasteful access time to the buffer memory may be eliminated to improve the data transfer efficiency.

Buffer Memory Map

Referring to Fig. 9, each sector of the buffer memory 2 has a capacity of 8 k(8192) bytes, of which 6 k(6144) bytes represent a usable area. The area shown by hatched lines is not pertinent directly to the present invention and therefore the related description is omitted for simplicity. The 128-frame or one-sector coding data is written in the left half 32x128 byte portion in the figure. The C_2 and C_1 parity codes are written in the 4x128 byte area adjacent to the area where the coding data are written. The C_1 correction flags dependent on the results of the error correction decoding by the C_1 parity codes and the read flag indicating the data write time are written in the 1x128 byte area at the right-hand side of the figure. The read flags and the C_1 correction flags are set for each frame, with the read and C_1 correction flags for each frame being written in the same 8-bit byte. The frame address is written in the 1x128 byte area adjacent to this area. The sub-codes S_{C0} to S_{C3} having the same contents as described above are written in the 1x128

byte area adjacent to the area where the frame address is written. In the figure, the upper 4-byte area of the 1x128 byte area adjacent to this area is reserved for the correction flags by the parity codes of the sub-codes, while the lower 7-byte area is used as the internal register for the ECC processor 13.

The read flags and the C_1 correction flags will be described in detail. As shown to an enlarged scale in Fig. 10A, the reading flags and the C_1 correction flags for each frame are written in the 1-byte or 8-bit area in the buffer memory 11. The upper five bits D_3 to D_7 and the three bits D_0 to D_2 represent the area for the read flags and the area for the C_1 correction flags, respectively. As shown in Fig. 10B, when the data are written into the buffer memory 2, all the bits are "1". When the error correction and decoding is performed by the C_1 parity codes, the upper 5-bits for reading flags are all "0", while the lower 3 bits for C_1 correction flags are "1" or "0" depending upon the results of correction and decoding. Thus they are "000" when there is no error, "001" when one error is to be corrected, "011" when 2 errors are to be corrected and "111" when the correction is impossible. In other words, a flag is set at F_0 bit in case of one error correction, flags are set at F_0 and F_1 bits in case of two error correction, and flags are set at the three bits F_0 , F_1 and F_2 in case the correction is not possible.

It is noted that the above C_1 correction flag is intended for error correction by the C_2 parity codes as later described.

In this manner, the memory area for the reading flags and that for the C_1 correction flags are provided in one and the same byte so that only one byte suffices for the flag area for each frame. Since the upper five bits for the reading flags are all "0", that is, reset automatically, at the time of error correction by the C_1 parity codes, it becomes unnecessary to reset the reading flag again for reproducing the next sector.

These flag states are transferred as the occasion may require to the aforementioned status register 22.

Error Correction

A typical method for correcting the error in the data frame and sub-code will be explained.

Error Correction for Data Frame

An algorithm of error correction of the data frame will be explained first by referring to Fig. 11. At step 301, the data contained in the C_1 code of each frame are sequentially read from the buffer memory 2 and transmitted to the error correction processor 13. At step 302, the error correction processor calculates the syndrome using the C_1 parity codes,

and finds the number of the errors based on the syndrome status. It is then checked at step 303 whether the error is the 2 symbol error. If the result is negative, it is checked at step 304 whether the error is the 1 symbol error. If the result is negative, it is checked at step 305 whether there is no error or the error is the error of 3 or more symbols. If the error is determined to be the 1-symbol error at step 304, the error location is found at step 306 using the syndrome, and the error symbol is corrected. If the error is determined to be the 2 symbol error at step 303, the error location is found at step 307 using the syndrome and the error symbol is corrected. At the next step 308, the flags F_1 of the correction flag area of the buffer memory 2 shown in Figs. 9 and 10A are set as indicated in Fig. 10B. In case of no error or of effecting 1-symbol or 2-symbol error correction, it is checked at step 309 whether continuity exists between the frame address of the frame and the frame address of the frame immediately before it. In case of discontinuity or in case the error of three or more symbols is found is found at step 305 so that correction is not possible, the flags F_0 , F_1 and F_2 are set at the next step 310 as indicated in Fig. 10 B.

The above steps are repeatedly executed for each frame. If it is found at step 311 that the processing of all of the frames is terminated, decoding of the next C_2 series is

performed. The error correction for the sub-frames is performed in the similar manner and in advance of the above described processing of the data frames.

In the error correction for the C_2 series, the data of the C_2 series interleaved from the buffer memory 2 are read out at step 312 and the syndrome is calculated at step 313. The number of errors is then determined from the syndrome status. It is then checked at step 314 whether the error is the 2-symbol error. If the result is negative, it is checked at step 315 whether the error is the 1-symbol error. If the result is negative, it is checked at step 316 whether there is no error or whether the error is the error of three or more symbols. If the error is found to be the 1-symbol error at step 315, the error location is found at step 317 and the error symbol is corrected. In more detail, referring to Table 1, when the error location is not coincident with the location of the symbol to which the F_2 flag has been afforded in the C_1 correction processing, it is checked by the number of the F_2 flags whether the correction is to be performed. If the correction is not made, the current state is maintained. In the Table 1, N_1 and N_2 denote the numbers of the symbols to which the flags F_1 and F_2 are afforded and that are coincident in locations to the error locations. k_1 and k_2 denote the numbers of the F_1 and F_2 flags and X means "don't care".

Table 2

error	flag condition	operate
0	—	no correct
1	$N2=1$ $N2=0 \quad K2 \leq 3$ $N2=0 \quad K2 \geq 4$	1 error correct 1 error correct uncorrect
2	$N1=2 \quad N2=2 \quad K2=2$ $N1=2 \quad N2=1 \quad K2 \leq 2$ $N1=2 \quad N2=0 \quad K2=0$ $N1=2 \quad N2=0 \quad K1 \leq 4 \quad K2=1,2$ $N1=2 \quad N2=0 \quad K1 \geq 5 \quad K2=1,2$ $N1=1 \quad K1 \leq 3 \quad K2 \leq 2$ $N1=1 \quad K1 \geq 4 \quad K2 \leq 2$ $N1=0 \quad K2 \leq 2$ $K1=3 \quad K2=3$ $K1 \geq 5 \quad K2=3 \quad (N2=1)$ $K1 \geq 5 \quad K2=3 \quad (N2=0)$ $K1=4 \quad K2=3,4$ $K1 \geq 5 \quad K2 \geq 4$	2 error correct 2 error correct 2 error correct 2 error correct uncorrect 2 error correct uncorrect uncorrect 3 erasure correct 2 erasure & 1 error correct uncorrect 4 erasure correct uncorrect
more than 3	$K1=3$ $K1 \geq 5 \quad K2=3 \quad (N2=1)$ $K1 \geq 5 \quad K2=3 \quad (N2=0)$ $K1=4$ $K1 \geq 5 \quad K2 \neq 3$ $K1 \leq 2$	3 erasure correct 2 erasure & 1 error correct uncorrect 4 erasure correct uncorrect uncorrect

$N1$: number of flags corresponding with the error location

$N2$: number of flags corresponding with the error location

$K1$: number of $F1$ flag

$K2$: number of $F2$ flag

$\times$: don't care

When the error is determined to be the 2-symbol error at step 314, the number of the F_2 flags is determined at step 318. When the error is the error of 2 or less symbols, the 2-symbol error correction processing is performed at step

319. When the error is the error of more than two symbols, an erasure correction processing as later described is performed. It is determined at step 319 whether the 2-symbol error correction should be performed under the condition shown in Table 2, or the data should be left as are without error correction.

When it is determined that there exist errors of three or more symbols at step 316 or the number of F_2 flags exceeds two at step 318, it is determined at steps 320 and 321 whether the number of F_1 flags is 3 or 4. When it is three, a 3-erasure correction is performed at step 322 and, when it is four, a 4-erasure correction is performed at step 323.

When it is determined at steps 320, 321 that the number of F_1 flags is neither 3 nor 4, the numbers of F_1 and F_2 flags are determined at step 324. A 2-erasure correction and a 1-error correction are performed at step 322 only when the number of F_1 flags is not less than 5 and the number of F_2 flags is 3.

When the correction is not made in the above procedure, the data are treated as error.

The above steps are repeated for each series and, when it is determined at step 325 that the processing of these series is terminated, the correction is terminated.

At the time of the correction by the C_2 parity codes, when it is determined above all that there exist errors of

two or more symbols, an erasure correction routine is performed when the number of F_2 flags set for symbols contained in the C_2 series exceeds 2, that is, 3 or more, and a 2-error correction routine or double error correction is performed when the above number does not exceed two. It is when the errors of 3 or more symbols or burst errors are caused that the F_2 flags are set in the course of correction by the C_1 parity codes. However, in a majority of cases, it is the burst errors that are caused, if the symbol error rate is not higher than 10^{-3} . In this manner, wasteful 2-error correction routines are not executed on occurrence of the burst errors and the uncorrectable errors otherwise caused during the correction routines may be avoided with a significantly improved correction capability for burst errors. The result would be most favorable when the burst error length is 8 to 14 frames.

It will be appreciated from the foregoing description that, in the present embodiment, when it is determined that there are caused errors of a number of symbols larger than the number that may be corrected for errors, such as two symbols, an erasure correction is or is not executed according as the number of flags indicating that the correction is not possible by the C_1 parity codes and that are set for symbols contained in the C_2 series exceeds or does not exceed a predetermined number, such that the

correction capability for burst errors may be improved significantly.

It is noted that up to double errors can be corrected by the C_1 parity codes, while up to quadruple errors can be corrected by the C_2 parity codes. In more detail, error correction of m symbols and/or erasure correction of n symbols are possible with the C_2 parity codes, wherein m and n are given by the formula

$$2m + n \leq d - 1$$

where the minimum distance between the codes $d=5$. In case of 4-symbol errors, the following formulas hold for the syndromes S_0, S_1, S_2 and S_3 .

$$\begin{cases} S_0 = e_i + e_j + e_k + e_l & \dots (1) \\ S_1 = \alpha^1 e_i + \alpha^1 e_j + \alpha^1 e_k + \alpha^1 e_l & \dots (2) \\ S_2 = \alpha^{2^1} e_i + \alpha^{2^1} e_j + \alpha^{2^1} e_k + \alpha^{2^1} e_l & \dots (3) \\ S_3 = \alpha^{3^1} e_i + \alpha^{3^1} e_j + \alpha^{3^1} e_k + \alpha^{3^1} e_l & \dots (4) \end{cases}$$

where e_i, e_k, e_k and e_l represent error patterns, represents the root satisfying the irreducible polynomial $F(x)=0$ on the galois field $GF(2^M)$, where, for example, $M=8$, and i, j, k and l represent error locations. In case of a 3-symbol error, $e_l=0$. The error locations i, j and k only are known in the triple erasure correction, while the error locations i and j only are known in the double error

correction plus single error correction.

Solving the simultaneous equations (1) to (4) with respect to the error patterns e_i , e_j , e_k and e_l , the following equations for the quadruple erasure correction are obtained.

$$e_i = \frac{S_0(\alpha^{-j} + \alpha^{-k} + \alpha^{-l})S_1(\alpha^{-j-k} + \alpha^{-k-l} + \alpha^{-l-j})S_2 + \alpha^{-j-k-l}S_3}{(1 + \alpha^{l-j})(1 + \alpha^{l-k})(1 + \alpha^{l-l})} \quad \dots (5)$$

$$e_j = \frac{S_0(\alpha^{-k} + \alpha^{-l} + \alpha^{-i})S_1(\alpha^{-k-l} + \alpha^{-l-i} + \alpha^{-i-k})S_2 + \alpha^{-k-l-i}S_3}{(1 + \alpha^{j-i})(1 + \alpha^{j-k})(1 + \alpha^{j-l})} \quad \dots (6)$$

$$e_k = \frac{S_0(\alpha^{-l} + \alpha^{-i} + \alpha^{-j})S_1(\alpha^{-l-i} + \alpha^{-i-j} + \alpha^{-j-l})S_2 + \alpha^{-l-i-j}S_3}{(1 + \alpha^{k-j})(1 + \alpha^{k-i})(1 + \alpha^{k-l})} \quad \dots (7)$$

$$e_l = \frac{S_0(\alpha^{-i} + \alpha^{-j} + \alpha^{-k})S_1(\alpha^{-i-j} + \alpha^{-j-k} + \alpha^{-k-i})S_2 + \alpha^{-i-j-k}S_3}{(1 + \alpha^{l-i})(1 + \alpha^{l-j})(1 + \alpha^{l-k})} \quad \dots (8)$$

In case of the triple erasure correction or the double erasure correction plus single error correction, the simultaneous equations (1) to (4) ($e_l=0$) are solved with respect to the error patterns e_i , e_j and e_k to give the following equations (9) to (12) wherein, in the double error correction plus single error correction, α^k , that is, the error location, is also found.

$$\alpha^k = \frac{S_1 \alpha^{i+j} + S_2 (\alpha^i + \alpha^j) + S_3}{S_0 \alpha^{i+j} + S_1 (\alpha^i + \alpha^j) + S_2} \quad \dots (9)$$

$$e_i = \frac{S_0 \alpha^{j+k} + S_1 (\alpha^j + \alpha^k) + S_2}{(\alpha^i + \alpha^j) (\alpha^i + \alpha^k)} \quad \dots (10)$$

$$e_j = \frac{S_0 \alpha^{i+k} + S_1 (\alpha^i + \alpha^k) + S_2}{(\alpha^j + \alpha^i) (\alpha^j + \alpha^k)} \quad \dots (11)$$

$$e_k = \frac{S_0 \alpha^{i+j} + S_1 (\alpha^i + \alpha^j) + S_2}{(\alpha^k + \alpha^i) (\alpha^k + \alpha^j)} \quad \dots (12)$$

In this manner, in the cases of quadruple erasure correction, triple error correction and double error correction plus single error correction, the arithmetic operations by separate equations are usually carried out, with an increased program volume and time involved in the arithmetic operations.

Thus the quadruple erasure correction and triple erasure correction (or double erasure correction plus single error correction) are carried out using the common terms for the two sets of equations, that is, the set of equations (5) to (8) for finding the quadruple error patterns and the set of equations (9) to (12) for finding the triple error patterns, instead of carrying out separate arithmetic operations using

these sets of equations. In the present embodiment, these common terms are expressed by the following equations.

$$\begin{cases} A = \alpha^{1+j} & \dots (13) \\ B = \alpha^1 + \alpha^j & \dots (14) \\ C = S_1 A + S_2 B + S_3 & \dots (15) \\ D = S_0 A + S_1 B + S_2 & \dots (16) \end{cases}$$

Using these equations (13) to (16) for the above common terms, the equations (15) to (18) for finding the quadruple error patterns may be expressed by the following formulas (17) to (19):

$$e_i = \frac{D \alpha^k + C}{(A + B \alpha^1 + \alpha^{21}) (\alpha^1 + \alpha^k)} \quad \dots (17)$$

$$e_k = \frac{D \alpha^1 + C}{(A + B \alpha^k + \alpha^{2k}) (\alpha^1 + \alpha^k)} \quad \dots (18)$$

$$e_j = \frac{S_0 \alpha^1 + S_1 + (\alpha^1 + \alpha^k) e_k + (\alpha^1 + \alpha^1) e_i}{B} \quad \dots (19)$$

$$e_i = S_0 + e_i + e_k + e_j \quad \dots (20)$$

Using the same equations (13) to (16), the equations (9) to (12) for finding the triple error patterns may be expressed by the following formulas (21) to (24):

$$\alpha^k = \frac{C}{D} \quad \dots (21)$$

$$e_k = \frac{D}{(\alpha^i + \alpha^k)(\alpha^j + \alpha^k)} \quad \dots (22)$$

$$e_j = \frac{S_0 \alpha^i + S_1 + (\alpha^i + \alpha^k)e_k}{B} \quad \dots (23)$$

$$e_i = S_0 + e_k + e_j \quad \dots (24)$$

The above common terms are found in advance and substituted into the equations (17) to (20) to find the error patterns e_i , e_j , e_k and e_l in the case of the quadruple error correction and into the equations (21) to (24) to find the error patterns e_i , e_j and e_k in the case of the triple erasure correction and double error correction plus single error correction. It is noted that α^k is to be found only in the case of the double erasure correction plus single error correction.

In this manner, when carrying out the arithmetic operations for error correction and/or erasure correction of different numbers of symbols, those portions common to these operations are computed in advance for reducing the program

volume for the arithmetic operations and shortening the computing time.

The methods of deriving the equations (17) to (20) for finding the quadruple error patterns and the equations (21) to (24) for finding the triple error patterns will be hereafter explained. First, the equations for finding the quadruple error patterns are derived in the following manner. By multiplying denominators and the numerators of the equations (8) and (7) by α^{i+j+k} and α^{i+j+1} , respectively, the following equations, we obtain

$$\begin{aligned}
 e_i &= \frac{\alpha^{i+j+k} S_0 + (\alpha^{i+j} + \alpha^{j+k} + \alpha^{k+i}) S_1 + (\alpha^i + \alpha^j + \alpha^k) S_2 + S_3}{(\alpha^i + \alpha^j) (\alpha^i + \alpha^j) (\alpha^i + \alpha^k)} \\
 &= \frac{\{\alpha^{i+j} S_0 + (\alpha^i + \alpha^j) S_1 + S_2\} \alpha^k + \{\alpha^{i+j} S_1 + (\alpha^i + \alpha^j) S_2 + S_3\}}{\{\alpha^{i+j} + (\alpha^{i+j}) \alpha^i + \alpha^{2i}\} (\alpha^i + \alpha^k)} \\
 &\dots (25)
 \end{aligned}$$

$$\begin{aligned}
 e_k &= \frac{\alpha^{i+j+1} S_0 + (\alpha^{i+j} + \alpha^{j+1} + \alpha^{i+1}) S_1 + (\alpha^i + \alpha^j + \alpha^1) S_2 + S_3}{(\alpha^k + \alpha^i) (\alpha^k + \alpha^j) (\alpha^k + \alpha^1)} \\
 &= \frac{\{\alpha^{i+j} S_0 + (\alpha^i + \alpha^j) S_1 + S_2\} \alpha^1 + \{\alpha^{i+j} S_1 + (\alpha^i + \alpha^j) S_2 + S_3\}}{\{\alpha^{i+j} + (\alpha^{i+j}) \alpha^k + \alpha^{2k}\} (\alpha^k + \alpha^1)} \\
 &\dots (26)
 \end{aligned}$$

By introducing the above equations (13) to (16) for the

above common terms, the equation (17) and (18) are obtained from the above equations (25) and (26). Also, by finding the equation (1) $\times \alpha^i$ plus equation (2), the following equations.

$$\alpha^i S_0 + S_i = (\alpha^i + \alpha^j) e_j + (\alpha^i + \alpha^k) e_k + (\alpha^i + \alpha^l) e_l \quad \dots (27)$$

$$\therefore e_j = \frac{\alpha^i S_0 + S_i + (\alpha^i + \alpha^k) e_k + (\alpha^i + \alpha^l) e_l}{\alpha^i + \alpha^j} \quad \dots (28)$$

are obtained. By introducing the equation (14), the equation (19) may be obtained from the equation (28). Also the equation (20) may be obtained from the equation (1).

The equation for finding the triple error patterns are derived in the following manner. The equation (21) and (22) may be obtained by introducing the above equations (13) to (16) for the common terms into the equations (9) and (12). Also, by setting $e_l = 0$ in the equation (28), we obtain

$$e_j = \frac{\alpha^i S_0 + S_i + (\alpha^i + \alpha^k) e_k}{\alpha^i + \alpha^j} \quad \dots (29)$$

so that, by introducing the equation (14), the equation (23) may be obtained from the equation (29). The equation (24) may be obtained by setting $e_l = 0$ in the equation (1).

It will be appreciated from above that, according to the error correction and decoding method of the present invention, those portions common to the arithmetic operations for the error correction and/or erasure correction with

different numbers of symbols are computed in advance, such that the program volume and the computing time necessary for those arithmetic operations may be reduced.

Attention is directed to our copending application
5 8812941.6 (GB-A-2,205,423) from which this application was divided and also to our copending applications: ~~9106141.6~~
~~(GB-A-2,241,803)~~; 9106142.4 (GB-A-2,241,804) and 9106143.2 (GB-A-2,241,805) which were also divided thereon.



CLAIMS

1. An apparatus for receiving digital data encoded in an error correction code and performing an error correction decoding, comprising:

5 a first memory area for storing received data,

a second memory area for storing readout flag data during writing of said received data into said first memory area,

processing means for performing said error correction decoding on said received data, and

a third memory area for storing flag data corresponding to a result of said error correction decoding, wherein

said second memory area and said third memory area are provided in the same byte within a single memory,

said second memory area and said third memory area are set from a first value to a second value during writing of said received data in said first memory area, and

said second memory area is set to said first value during said error correction decoding of said received data while said third memory area is set to a value corresponding to the result of said error correction decoding.

2. An apparatus according to claim 1, wherein said error correction code comprises a Reed-Solomon parity code.



3. An apparatus according to claim 1 or 2, wherein said first memory area is also provided in the same memory as said second and third memory areas.

4. An apparatus according to claim 1, 2 or 3, wherein
5 the information data includes first error correction codes formed from a first series of data, and second error correction codes formed from a second series of data which are interleaved with the first series of data, and said means for performing said error correction decoding comprises:

10 first decoding means for performing decoding of respective first error correction codes;

means for generating flag data indicating error correction impossible with regard to respective first series of data when correction of erroneous data included in the
15 respective first series of data is impossible;

second decoding means for performing decoding of respective second error correction codes, and comprising:

means for determining the number of erroneous data included in the respective second series of data;

20 means for correcting erroneous data included in the respective second series of data when the number of erroneous data is determined to exceed a first predetermined number and the number of said flag data relating to the data of said second series of data does not exceed a second predetermined



number;

means for erasure correcting erasure data included in the respective second series of data when the number of erroneous data is determined to exceed a first predetermined number and the number of flag data added to the second series of data exceeds a second predetermined number.

5. An apparatus according to claim 1, 2 or 3, wherein the information data is encoded with respective error correction codes having a minimum code distance equal to d , where d is an integer, and wherein the means for performing said error correction decoding decodes respective error correction codes satisfying the equation;

$$2m + n \leq d - 1$$

where, m is a number of erroneous symbols to be corrected and n is a number of erasure symbols to be corrected;
