I. B. Oldham*
R. T. Chien†
D. T. Tang§

# Error Detection and Correction in a Photo-Digital Storage System

**Abstract:** An error-correction system has been implemented for data stored in the IBM Photo-Digital Storage System. Hardware is used for encoding and error detection, and a processor-controller is used, on a time-sharing basis, for error correction. A Reed-Solomon code is used to obtain a very low error rate in spite of flaws affecting the recorded bits. This approach is applicable to systems which require complex codes and have a data processor available on a time-sharing basis.

## Introduction

In a high-density photo-digital storage system, contamination and other defects can easily obliterate a group of data bits. To operate successfully in spite of this problem in the IBM Photo-Digital Storage System[1,2] (referred to here as PDSS) a powerful error-correction code is used.

Information obtained on the operation of a predecessor to the PDSS, the Photostore System,[3] indicated that the desired net error rate of one bad line of data per $8.1 \times 10^8$ bits read might be met if a Reed-Solomon[4] code with 50 six-bit characters of data and eleven characters of redundancy were used. The difficulty with using a code of such complexity is that implementation of the decoding system in hardware with adequate speed is very complex, and implementation with a sequential processor is too slow.

The problem of effectively implementing a code of this complexity has been solved by a number of innovations. Most important is the use of hardware for encoding, calculation of the power sums, and error detection, while using a control processor, on a time-sharing basis, for error correction. Another important feature is that single-character error correction is tried first; if this is not sufficient, further correction activities can be tried. Other important features are use of a "trial and recheck" method of error correction, selection of a symmetric code polynomial, use of a table of logarithms for multiplication and division in a Galois field of 64 elements, et cetera.

The approach used here differs from that of Bartree and Schneider[5] in that limited additional circuitry is used with a standard data processor rather than building a special-purpose processor. The procedure used for error correction follows the basic method of Gorenstein, Zierler and Peterson,[6,7] with modifications to improve the speed of decoding. Two of the modifications are the omission of the step to determine the number of errors, and the inclusion of direct solution of quadratic equations in a Galois field of 64 elements.

## Coding requirements for a photo-digital memory system

To permit better understanding of the coding requirements and the choice of features used in implementing the code, the PDSS will be described briefly.

Data entering and leaving the storage system are handled by a data controller which encodes the lines of data, does error detection, and contains adequate buffering to hold a line of data while error correction is being done on the line. All encoding and error detection, as well as some correction functions, are implemented by circuitry in the data controller.

Data are recorded on 1.377″ by 2.750″ photographic film "chips," using an electron beam recorder to expose the film.[8] Each bit occupies an area 14$\mu$ by 16$\mu$. The bits are written sequentially in lines of 300 data bits, together with redundancy and format bits. The film chips are developed on

* I. B. Oldham is at the Systems Development Division Laboratory of the International Business Machines Corporation, San Jose, California.
† R. T. Chien is at the Coordinated Science Laboratory of the University of Illinois, Urbana, Illinois, and is a consultant to the International Business Machines Corporation.
§ D. T. Tang is at the T. J. Watson Research Center of the International Business Machines Corporation, Yorktown Heights, New York.

line and stored in small boxes, called "cells," which are stored in file modules. The data can then be read optically with a flying-spot scanner.

Various decision-making functions of the system, including control of the film-handling mechanisms, are performed by a stored-program control processor. This processor also does most of the calculations necessary for error correction. Time sharing of the processor is accomplished by the use of multiple interrupt levels.[9]

With adequate buffering between the storage system and the main computing processors, delays associated with error correction are tolerable, provided the average data rate remains high. This buffering also allows newly recorded chips to be checked for readability and to be rerecorded, if necessary, with data kept in the buffer.

Now let us consider the coding requirements for this system. In a photo-digital system, errors are caused by contamination, distortion of the emulsion, or other problems affecting the quality of the data. These flaws may be introduced during manufacturing and transporting of the raw chips, during recording and development of the chips, or during subsequent storage and reading.

To prevent loss of data, some errors which occur can be corrected using an error-correcting code. Some errors can be eliminated by the use of a readability check which is done immediately after recording; unreadable data is rerecorded. Neither approach alone suffices because coding cannot overcome flaws which obliterate a large part of a line. Rejecting and rerecording alone would not be sufficient because virtually all chips have minor flaws affecting at least a few bits, and because it does nothing to help the problem of contamination during subsequent storage.

To make an effective selection of a code for use in the IBM PDSS, the Photostore (an earlier system which has somewhat similar characteristics) was studied. The Photostore System is a photographic system which originally wrote circular tracks of data with an optical recording arrangement.[3] The recording and reading characteristics were somewhat different from those planned for the PDSS, but it was reasonable to expect that the Photostore would be a fair model for the PDSS.

The procedure used in selecting a code was to read a large quantity of known data from the Photostore and record all errors in the raw data. The known capability of various codes was tested empirically by Chien, Tang, Barrekette, and Katcher,[3] using blocks of data that were read. For example, a code which was known to correct all single bursts of length 17 or less within a block of data was judged to have successfully decoded a given block of data if these were, in fact, errors occurring only within one burst of length 17 or less. By comparing burst correcting codes, independent bit-correcting codes, and independent character correcting codes, it was determined that an independent character correcting code with 11 characters of redundancy, a block length of 63 characters, and a character size of 6 bits best met the objective of meeting the required error rate while minimizing the required redundancy. This type of code is known as a Reed-Solomon code[4] or Bose-Chaudhuri-Hocquenghem character correcting code.[6,10,11]

For selected good tracks on selected disks, the Photostore System demonstrated a raw error rate of 1 block per $8 \times 10^4$ bits read, for 441 bit blocks. The reliability required for the IBM PDSS, after error correction, was a rate of 1 erroneous line for every $8.1 \times 10^8$ data bits read and 1 undetected error for every $2.7 \times 10^{10}$ data bits read, for lines with 300 data bits. The character correcting code selected showed promise of being able to meet these requirements. It is effective against both random flaws of a few bits each and bursts of moderate length caused by a larger flaw.

Having chosen the type of code, the remaining problem was to select the exact generator polynomial[6] for the code. The generator polynomial selected was

$$\begin{aligned} g(x) &= \prod_{i=-5}^{5} (x - \alpha^i) \\ &= x^{11} + \alpha^{14}x^{10} + \alpha^{59}x^9 + \alpha^6x^8 + \alpha^{28}x^7 \\ &\quad + \alpha^{54}x^6 + \alpha^{54}x^5 + \alpha^{28}x^4 + \alpha^6x^3 \\ &\quad + \alpha^{59}x^2 + \alpha^{14}x + \alpha^0, \end{aligned} \tag{1}$$

where $\alpha$ is a primitive element in the Galois field of $2^6$ elements formed by all polynomials modulo $x^6 + x + 1$, with coefficients in the Galois field of 2 elements.

There were two reasons why this particular generator polynomial was chosen. One was that $\alpha^0$ is one of the roots. Thus, one of the power sums is a longitudinal parity check, which means that one of the power sums will be equal to the magnitude of the error when a single error occurs. This property is necessary in order to correct single errors rapidly. The second property which this particular polynomial possesses is symmetry; thus, when it is written in its expanded form, there are five distinct coefficients rather than ten. This results in hardware saving in the encoder, because the encoder contains circuits which multiply a given input by each of the distinct coefficients of the generator polynomial.

## Implementation approach

Encoding and calculation of power sums for the Reed-Solomon code are done in hardware. Other portions of the decoding activity are implemented in software in a control processor which is time-shared with the control functions and other functions of the storage system. This processor is interrupted when an error is detected during reading.

By providing programs which do most of the computations necessary for error correction within the control processor, a relatively complex decoding procedure can be implemented with minimum additional hardware beyond that

required for data buffering, which is needed for data-rate matching. This makes it possible to have a system which operates rapidly in hardware when there are no errors and which allows complex error-correction calculations in software with little requirement for additional hardware.

Without this division of functions, it would not be feasible to implement a code of the complexity used. Because the IBM PDSS is a very large storage system which is buffered from the main processing units of the system, occasional long decoding times are tolerated, provided that the average decoding time is short. This is a freedom which is often not available in data processing storage systems.

For reasons of speed, encoding is also implemented in hardware. The encoding function is performed by a circuit capable of dividing the input line of data by the generator polynomial. The error detection function could be performed by this same circuit or, in order to allow reading and writing to occur at the same time, by a duplicate circuit. Instead, the error detection function is accomplished by alternative circuitry. The error detection circuitry divides by the factors of the generator polynomial rather than by the entire polynomial itself. These check circuits have no advantage or disadvantage as error detecting circuits. However, when an error is detected, the check circuits produce the power sums which are used by the error correcting procedure.

By transmitting the power sums to the control processor and interrupting it, the error correction program can begin the correction immediately. The processor is shared with the control functions which may interrupt the error correction process during long error correction computations. This method of operation makes the relatively powerful processor available without unduly interfering with its other function of control, and makes implementation of the powerful independent character correcting code feasible.[9]

The fastest available decoding techniques are used first. Slower, more powerful, techniques are not attempted until the faster ones have been exhausted. This decoding strategy is designed to minimize the average decoding time at the expense of increasing the maximum decoding time.

Initially, single-error correction is used to try to correct an error of a single character. This is usually sufficient because most errors affect only a single character. If it is not sufficient, the line is reread. In this system, single-error correction occurs fast enough that it can be done while the reader scans back to the beginning of the line to reread the line.

After trying single-error correction and rereading several times, two-error correction is attempted; then three-error correction is tried if two-error correction is not adequate. Other slower corrective actions can be tried singly or in combination until all have been exhausted before abandoning a line. These slower actions include correcting up to five errors and other data recovery techniques. To exhaust all these techniques takes a long time; it is seldom necessary, however, to use most of them, so the effect on the average throughput is small in spite of the very long time required for some bad lines.

The error correction code is limited to correcting additive errors. That is, if the beginning of the line is known, if most of the bits can be read, and if the bits which have been successfully read can be correctly positioned on the line, then the code can correct some characters that are in error. This leaves several classes of difficulties which are not specifically correctable by the code. They include loss of synchronism, inability to find the start of the line, and errors in the line numbers that make it difficult to know what line to read.

While the code does not directly help in solving these problems, it plays a very important part in the procedures which are used in attempting to overcome them. Namely, it can effectively reject a combination of bits which does not make up a correct, properly synchronized line. This makes it possible to make a number of reasonable guesses as to where the correct line is or where the beginning of a line is, with little fear of erroneously accepting the wrong thing as the line being sought. In this system the procedures which are used to overcome these kinds of problems are referred to as *data recovery procedures*.[12] They include rereading, searching for the correct line, jumping to the line from some other known line, physically moving the film chip to clean and reposition it, varying the reference point of the line-following servo in the reader, using a different method to find the beginning of the line, et cetera.

Besides the ability of the code to reject patterns of bits which are not the correct line, two other features of this system aid in the rejection of bad lines. One of these features is the inclusion of a 12-bit line number. If the system reads the wrong line, it can be rejected because the line number is not the one being sought. Any collection of bits which occurs due to improper synchronism of a line will have some group of bits other than the line number in the line number field and will thus usually be rejected as having the wrong line number. A second feature of the system is that three widely separated bits in the redundancy field are inverted as they come out of the encoder. They are reinverted when the data are read. If the line is properly read, these actions cancel each other out. If the line is out of synchronism when it is read, there will be six errors. This helps reject lines which are started six bits too soon or too late and which, if the bits in the line number field happened to be the correct line number, would otherwise be accepted as a valid coded line because all cyclic shifts of a coded line are also coded lines.

Some chips have gross faults which are present at the time data are recorded. These chips are eliminated by a readability check before storing them, to allow for rerecording. The readability check consists of reading the data

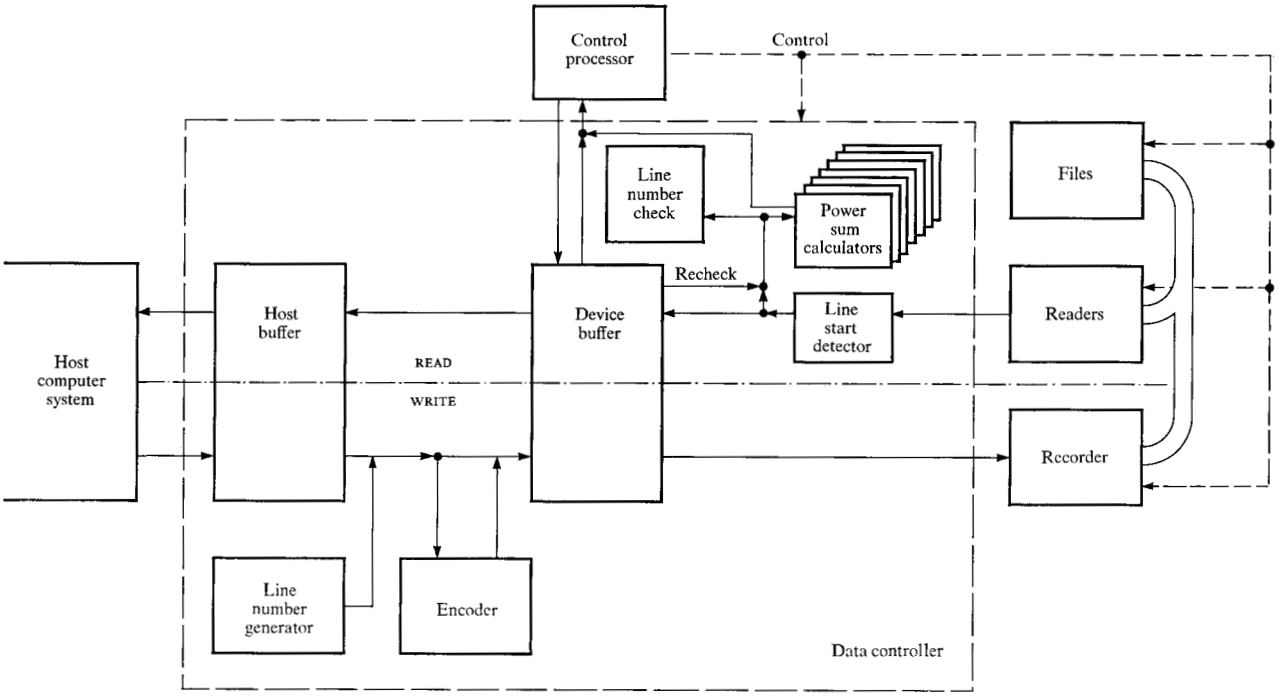



**Figure 1** Data flow in the data controller

from the reader into the data controller and verifying that it is readable; the data is not sent from the data controller to the host computer. During the readability check, the full capability of the data recovery procedure is not used, to prevent the acceptance of marginal chips which might otherwise cause reading problems subsequently.

The three techniques of error correction, data recovery, and readability check, taken together, made possible the very low error rate obtained.

### Error detection and correction system

During recording, the data are encoded by the data controller hardware. After recording, a readability check is made. During reading, the error detection and buffering of the data are done by the data controller hardware. Error detection is accomplished by computing the power sums of the line of data read. If the sums are not zero, a detectable error has occurred and error correction is attempted. Computation for this correction is done in the control processor. The correction is made to the data stored in the data controller buffer and the line is rechecked in the data controller. Logical decisions relative to how many errors to try to correct or what other procedures should be undertaken to obtain a correct line are made by the control processor. Flow of data in the data controller and in other parts of the system is shown in Figure 1.

For recording, the data to be recorded and line number characters are transmitted to the encoder on the input line as characters consisting of six bits in parallel. Incoming data to be recorded are broken up into 300-bit lines, to which the data controller appends a 12-bit line number. These data lines are treated as 52 six-bit characters which are considered to be elements of the Galois field of 64 elements. Each character is understood to be the coefficient of $x^i$ where $i$ equals 0 for the last character in the line, 1 for the next to last, etc., and $x$ is a dummy variable.

The encoder, shown in Figure 2, divides the line, treated as a polynomial, by the generator polynomial of the code. Each line and each block in this figure handles data six bits in parallel. The multiplier "$x\alpha^1$" is shown in Figure 3. To obtain a multiplier "$x\alpha^i$," use $i$ "$x\alpha^1$" multipliers in series and use logical simplification to minimize the circuitry. In fact, all the multipliers in the encoder are implemented in one large circuit with resultant saving of components.

After the 52 characters have been transmitted, the encoder contains the 11 remainder coefficients. This remainder is read out of the encoder and appended to the line to make a 63-character encoded line. After this encoding process the "coded-line polynomial" is divisible by the generator polynomial. Format bits for synchronizing the data are then added, making a total of 420 bits which are recorded as a line of data.

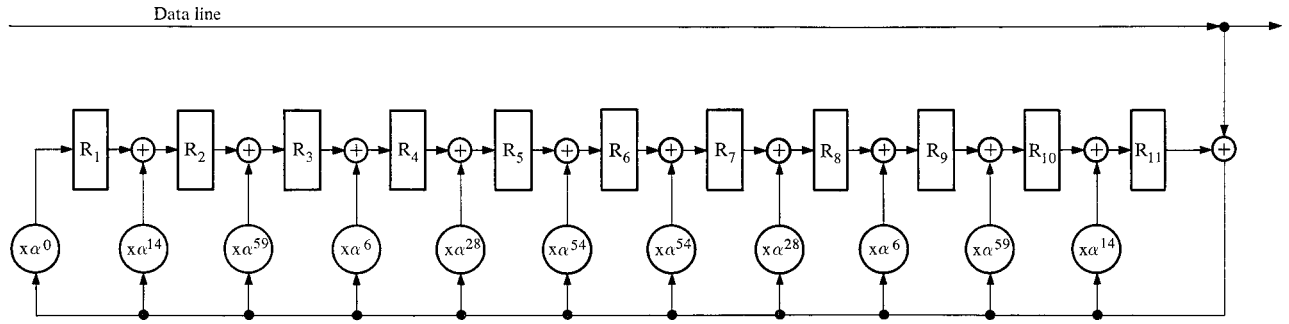
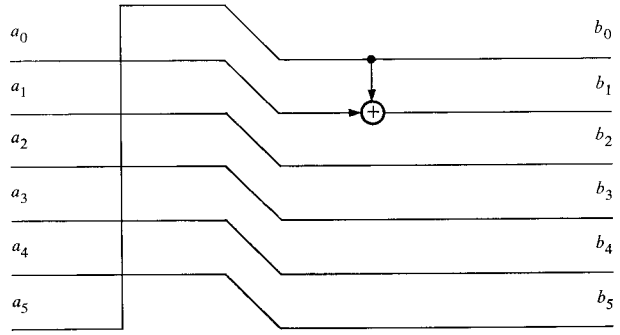



**Figure 2** Encoder

**Figure 3** Multiplier $x\alpha^1$ in which $b = \alpha^1 a$




The encoding process can be represented by the following equation:

$$A(x) = x^{11}D(x) - R(x), \tag{2}$$

where $A(x)$ is the coded line polynomial,

$D(x)$ is the data polynomial after the line number has been appended but before encoding, and

$R(x)$ is the remainder obtained when $D(x)$ is divided by $g(x)$, the generator polynomial given in Eq. (1).

After each chip has been recorded and developed, a check is immediately made to see if all the lines on it are readable with no uncorrectable errors. Any chip (or alternatively any record) which contains a line or lines that cannot be read and corrected must be rewritten. This procedure eliminates many potential errors. During the process, the full data recovery procedures of the reading system are not exhausted, to ensure the readability of the chip at a later date under slightly different conditions. After the chips in a cell have been checked and accepted, the cell is stored in a file module.

To read data, the cell containing the desired chip is withdrawn from the file module, and the chip is removed from the cell and positioned in the reader. While each line is read, the line number is verified, and a check is made to determine if the line is still divisible by the generator polynomial. These two tests are made in hardware and, if both criteria are met on a line which has been read or on which error correction has been attempted, the line is accepted and the reading proceeds. The test to determine if the line is still divisible by the generator polynomial is to divide the line by each of the 11 factors of the generator polynomial to see if the line is divisible by all of them. The 11 circuits used to make the tests are like the dividing circuit given in Figure 4, where $-5 \leq i \leq 5$. The 63 6-bit characters of the encoded line are fed into these circuits in parallel. Values obtained with these circuits are called the check sums or power sums. If some or all of the 11 values computed are not zero, an error has occurred in the line which was read and the power sums can be transmitted to the control processor for use in the error correction computations.

Two power sums are required per error to be corrected. Initially, the system does single error correction; thus, two power sums are transmitted to the control processor. The location of a single error is found by the control processor using the equation

$$L = 62 - (\log_\alpha S_1 - \log_\alpha S_0) \tag{3}$$

The address $L$ is transmitted to the data controller which adds, in Galois field arithmetic, $S_0$, the longitudinal parity, to the data character at the specified address. The line is then rechecked and if there was, in fact, one error, the line is accepted. If there was more wrong with the line than just a single-character error, rereading and further error correction is required.

The encoded lines of data are written in alternate directions, so the beginning of each line is just below the end of the previous line. To reread a line which has an error, the reader scans to the end of an adjacent line to get back to the beginning of the line to be reread. When a line that has been read contains an error, single-character error correction can take place while the reader is scanning an adjacent line to

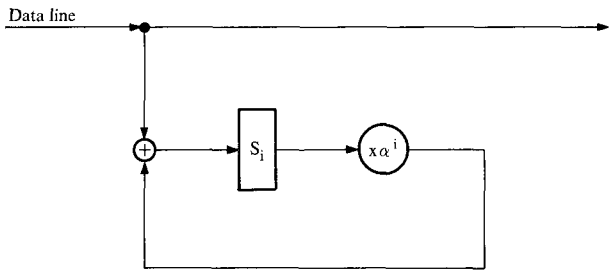



**Figure 4** $S_i$ calculating circuit

get back to the beginning of the line in error. If the line cannot be corrected by single character correction, the reader is ready to reread the line. If, after several tries of rereading and single-character error correction a line is not corrected, the reader can scan (without reading) the line of interest and an adjacent line in an idle loop while longer error correction processes are carried out.

After the control processor decides that single error correction and rereading have been used to the point of diminishing returns, multiple error correction begins. When the line is reread in the multiple error correction mode, ten power sums are transmitted from the data controller to the control processor. These sums are used to compute the location and magnitude of the errors.

The multiple error correction activity is used to try to correct two or fewer characters in error. If this is unsuccessful, the procedure allows up to three characters to be corrected, later up to four characters, and finally up to five characters.

On command from the control processor, the individual characters in error are transmitted from the data controller. In the control processor the magnitude of the error is subtracted in Galois field arithmetic, and the corrected character is transmitted back into its place in the buffer of the data controller.

After all of the corrections have been made, the line must be rechecked by recomputing the power sums. If they are all zero and the line number is correct, the line is considered to be corrected. If they are not zero, the line is reread and error correction is continued.

Several features of the program used for correcting the errors will now be discussed.

The power sums computed by hardware for a line that contains an error are used to solve for the elementary symmetric functions. These in turn are used to solve for the error locations which, in turn, are used together with the power sums to find the magnitudes of the errors. This procedure is basically that developed by Gorenstein, Zierler, and Peterson,[6,7] but with important modifications. Their procedure has five steps, namely, the calculation of (1) the power sums, (2) the number of errors, (3) the elementary symmetric functions, (4) the error locations, and (5) the error magnitudes.

One feature of our program is the assumption of the number of errors rather than solving for the number. The usual procedure is to consider all the possible equations relating the power sums to the elementary symmetric functions and determine the number of dependent equations. Then the independent equations are solved to obtain a correction which will make all the power sums zero.

In the procedure used here, no attempt is made to determine the number of characters in error. Instead, it is assumed that only one character is in error and thus single error correction is required. This assumption results in much faster decoding for those cases that can be corrected by single error correction or single error correction together with rereading. These are by far the most common cases. Similarly, if single error correction has failed, the process can then be continued by progressively assuming two, three, four, and five errors.

During any correction attempt, the actual number of errors may be different from the number of errors assumed. It is not determined in advance whether there is a solution for the errors or, if a solution does exist for the $2t$ error sums being considered, whether the solution will cause all eleven of the power sums to be zero. In fact, the process which attempts to find the errors may be terminated due to inability to find a solution for the error locations from the elementary symmetric functions. Thus it may be determined during the course of the attempted solution that there is no solution for (1) the number of errors being sought or (2) fewer than the number of errors being sought.

The solution produced, if any, has been calculated in such a way as to force $2t$ of the power sums to become zero, but not necessarily all eleven power sums. Thus, the proposed solution may satisfy the limited number of equations being solved, but not constitute a valid correction. After the correction has been made, the line must be rechecked to see if all the power sums are zero and the line has been corrected, or to see if some of the power sums are not zero, in which case the line has not been corrected. This is done by rechecking the line with hardware.

If there are fewer errors than assumed, there will be fewer independent equations relating the elementary symmetric functions to the power sums. When a dependent equation is found, it can be omitted and the number of power sum symmetric functions can be reduced to correspond to the actual number of errors. Then the solution can continue to find the smaller number of errors.

The advantage of this procedure is the quick solution of the simple set of equations obtained when a number of errors assumed is small. Thus, when a line has a small number of errors, or when a line can be corrected by rereading

and correcting a small number of errors, the error correction procedure is much faster than if an attempt were immediately made to correct up to five characters. The procedure is much faster on the average, inasmuch as most errors affect one or sometimes two characters when first read or subsequently reread. On the other hand, if the line is consistently read with five characters in error, the decoding procedure is slower because the unsuccessful attempts to correct less than five characters in error preceded the five-error correction. This means the average decoding time is substantially decreased and the maximum decoding time is substantially increased.

Besides the decrease in average decoding time, there is another advantage in that the line can be quickly reread many times. This allows lines with data of marginal quality to be read correctly, which might not otherwise be possible.

A problem associated with rereading a line which produces variable erroneous data is that each time it is read there is a possibility that the line will be accepted either before or after correction with an undetected error. The undetected error rate is, however, much lower when the number of errors sought is chosen to have a value less than five. Thus, rereading while attempting to correct one, two or three characters in error has negligible effect on the undetected error rate. The undetected error rate is little affected by those cases in which four errors are sought and is determined almost entirely by the number of times the line is read with attempts to do five-error correction. Thus, the undetected error rate is kept low in spite of the very large number of rereads allowed on any one line, because in most of the reads the number of errors sought is less than five.

Other unique features of interest include the following. In solving the polynomial which has as coefficients the elementary symmetric functions and has as roots the error locations, the use of a general solution for quadratics in a Galois field of even order speeds up the procedure. If the polynomial is of order one or two, the solution is obtained directly. If it is of higher order, the roots are sought by trial and error using synthetic division. After a root is found, the reduced polynomial, obtained by dividing the polynomial by the linear factor containing the root, is used for solving the remaining roots. When the order of the polynomial is reduced to two, the solution is obtained directly.

For all operations in the control processor, multiplication or division is done by adding or subtracting logarithms to the base alpha modulo 63, where alpha is a primitive root of the generator polynomial of the Galois field. This allows efficient computation in a Galois field of $2^6$ elements with a general-purpose computer.

Thus, in single error correction, the first power sum $S_1$ is divided by the $0^{th}$ power sum $S_0$ by looking up their logarithms in a table and subtracting them. Then, 62 minus the logarithm of the resulting error location is transmitted to the data controller. This format is chosen for the transmission of the error location inasmuch as it constitutes the address of the character in the buffer in the data controller.

In multiple-error correction, the elementary symmetric functions are obtained from the power sums using a matrix-solving program. The error locations are next obtained from the elementary symmetric functions by trial and error using synthetic division and by direct solution of the quadratic. The error magnitudes are then found from the error locations using the matrix solving program. Inasmuch as the decoding algorithms are implemented primarily in software, the program can be updated with current decoding procedures.

Not all errors can be corrected by algebraic error correction alone. In some cases the wrong line is read, the start of the line has not been found, or the reader may require an unusual adjustment in order to be able to read a particular line which is in error. The functions used to overcome these problems are collectively referred to as *data recovery.*[12]

Error correction and data recovery functions include:

1. Adaptively searching for the correct line by comparing the line number sought with the line number read, and then reading the next following or next preceding line to move toward the line sought;
2. correcting $t$ or fewer errors with algebraic error correction, where $t$ can be given any value from one to five;
3. rereading the line;
4. estimating the starting point of a line from timing if the start pattern cannot be found;
5. jumping several lines from the line currently being read to the correct line;
6. modifying the characteristics of the reader to cause it to read higher or lower on the bits or to cause its clock to change frequency faster, etc.;
7. repositioning the chip picker to clean the chip with an air brush and to back up to an area which has already been passed.

The fastest and most probably successful of the techniques is used first. It is followed by progressively slower and less likely techniques or combinations of techniques. This continues until the process is successful or all techniques are exhausted and a decision is made that the line cannot be corrected. Because of the power of this exhaustive data recovery procedure, almost all lines can be corrected.

## Results

The design error rate for the system was a maximum of one line detected in error per $8.1 \times 10^8$ bits read. The design criterion for undetected errors was a maximum of one line per $2.7 \times 10^{10}$ bits read. Both specifications were met.

The system operates satisfactorily within its designed average speed of $3.5 \times 10^5$ bits read per second while reading a representative sample of data records using one reader. Execution time of the single-error correction procedure is

**Table 1** Time required for various error-correction activities.

| | *Lines read* | *Single-error entries* | *Two-error attempts* | *Three-error attempts* | *Four-error attempts* | *Five-error attempts* | *Read recovery attempts* | *Picker repositioning* | *Total EDC & read recovery attempts* |
|---|---|---|---|---|---|---|---|---|---|
| Total occurrences | 108,366,272 | 849,085 | 5,341 | 3,370 | 215 | 190 | 15,315 | 48 | 873,516 |
| Time per occurrence in ms | 0.168 | 0.336 | 3.55 | 16.15 | 24.5 | 36.5 | 1.0 | 225 | 0.454 |
| Total time in ms | 18,200,000 | 285,000 | 18,900 | 54,600 | 5,280 | 6,940 | 15,000 | 10,800 | 397,000 |
| Fraction of total correction time | | 0.719 | 0.048 | 0.138 | 0.013 | 0.018 | 0.038 | 0.027 | 1.000 |
| Fraction of total scan time | 1.000,000 | 0.015,600 | 0.001,040 | 0.002,980 | 0.000,282 | 0.000,381 | 0.000,822 | 0.000,591 | 0.021.800 |

**Table 2** Success on steps used in error-correction activities.

| | *Total lines corrected* | *Success on try:* 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | *All other* | *Successes on picker reposition* | *Detected errors* | *Undetected errors* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Total | 783,856 | 768,939 | 9,002 | 1,071 | 333 | 165 | 106 | 76 | 284 | 267 | 1,174 | 1,481 | 958 | 29 | 8 | 0 |
| Fraction of corrections * | 1,000,000 | 981,000 | 11,500 | 1,360 | 425 | 211 | 136 | 97 | 362 | 341 | 1,500 | 1,760 | 1,220 | 37 | 10 | 0 |
| Fraction of lines read * | 7,260 | 7,100 | 83.3 | 9.9 | 3.08 | 1.52 | 0.98 | 0.70 | 2.63 | 2.47 | 10.8 | 13.7 | 8.87 | 0.27 | 0.07 | 0 |

* All values here have been multiplied by $10^6$.

about 0.34 milliseconds including time for rereading the line, if necessary. The two-error correction program requires about 3.5 ms, the three-error program 16 ms, the four-error program 25 ms, and the five-error program 37 ms. The processor used has 16-bit words, a 2-$\mu s$ cycle time, and a single address per instruction. The error computation program occupies a little less than 600 sixteen-bit words. Communication and other "overhead" related to the error correction functions occupy another 475 words.

The PDSS has demonstrated the ability to produce a net error rate about five orders of magnitude lower than the raw error rate without error correction. Simple, fast, decoding procedures were used 99.5% of the time. Slower decoding procedures were used occasionally. The net degradation of throughput due to error correction and read recovery was around 0.7%.

Tables 1 and 2 summarize the error correction activities in four 57-cell runs made at the time the second PDSS system was delivered to the Lawrence Radiation Laboratory at Berkeley.

Table 1 shows how much time is required for the various error correction activities. Note from the number at the bottom right of the table that error correction and data recovery require 2.18% of the scan time, where the scan time is defined as the time to scan the lines in the reader excluding the time required for changing columns, changing chips, changing cells, skipping between records, error-correction activities, etc. The fraction of the total system time used for error correction is a function of the fraction of total time spent scanning, which is a function of the application. For a 50-minute typical read test, 30.6% of the total time was spent scanning; thus, error correction and data recovery require 0.67% of the total time in this application. On the next-to-last line of Table 1, note that 71.9% of the time used for correction is used for single-error correction; in fact, about 70% of all correction time is spent on lines which are corrected immediately by the first attempt to correct a single error or by the first reread. Of the error correction and reread time, 13.8% is used for three-error correction, and 4.8% is used for two-error correction. The other 9.5% is used by other data recovery activities, picker repositioning, five-error correction, and four-error correction.

Table 2 shows the steps in the error correction and data recovery schedule that produced most of the corrections.

On trys 0 to 7, single-error correction followed by reread is used to attempt to correct the line. On try 8, two-error correction followed by reread is used. On try 9, three-error correction is attempted. Before rereading the data used in try 10, the system searches again for the correct line. Single-error correction is used in try 10.

Of all lines in error, 98.1% are corrected by try 0, and 99.8% are corrected within the first eleven tries. The last 0.2% of the lines in error are the hard ones to correct. The exhaustive data-recovery procedure is used to correct these lines.

The net detected error rate is $7.4 \times 10^{-8}$ lines in error per line read for chips which have been previously accepted as readable, which is appreciably better than the rate of $3.7 \times 10^{-7}$ required for delivery. The net detected error rate is about five orders of magnitude better than the raw error rate before correction of $7.26 \times 10^{-3}$ lines in error per line read.

The net undetected error rate is extremely small and

therefore difficult to measure with a reasonably sized sample. It is believed, from other observations which are available, that it is about two orders of magnitude better than the allowable rate of $1.11 \times 10^{-8}$ undetected lines in error per line read.

## Conclusions

The IBM Photo-Digital Storage System has demonstrated the effectiveness of hybrid hardware and software coding systems for use with relatively complex cyclic codes. This procedure is adaptable to any system which has a stored program processor available on a time-shared basis. Systems with this characteristic are becoming increasingly common. The processor may be a control or ancillary processor or may be the main processor for which data are being retrieved or may even be a processor whose main function might not be directly related to the storage or communications activity requiring coding.

This type of system offers powerful error-correction techniques and low average decoding time, but occasional long decoding times. The approach of hybrid hardware-software implementation shows promise of being an important implementation method for complex codes.


## Acknowledgments

A number of persons were involved in the design, building, and testing of the IBM PDSS. Some of those whose work was particularly relevant to this paper are B. D. Cunningham, programming; J. H. Davis and G. T. Moffitt, data controller; and R. L. Griffith, data recovery.


## References


1. J. D. Kuehler and H. R. Kerby, "A Photo-Digital Mass Storage System," *Proceedings of the Fall Joint Computer Conference*, p. 735, 1966.
2. Richard M. Furman, "IBM Photo-Digital Storage System," IBM San Jose SDD Technical Report 02.427, (1968).
3. R. T. Chien, D. T. Tang, E. S. Barrekette, and A. M. Katcher, "Analysis and Improvement of Photostore Error Rates," *Proc. IEEE* **56,** 805 (1968).
4. I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics* **8,** 300 (1960).
5. T. C. Bartree and D. I. Schneider, "An Electronic Decoder for Bose-Chaudhuri-Hocquenghem Error Correcting Codes," *IRE Trans. on Information Theory*, **IT-8,** S17 (1962).
6. W. W. Peterson, *Error-Correcting Codes*, The MIT Press, Cambridge, Massachusetts 1961.
7. D. Gorenstein and N. Zierler, "A Class of Error-Correcting Codes in $p^m$ Symbols," *Journal of the Society for Industrial and Applied Mathematics* **9,** 207 (1961).
8. K. E. Haughton, "An Electron Beam Digital Recorder," *Third International Electron and Ion Beams in Science and Technology Conference*, 1968.
9. D. P. Gustlin and D. D. Prentice, "Dynamic Recovery Techniques Guarantee System Reliability," to be presnteed at the 1968 FJCC.
10. R. C. Bose and C. R.-Chaudhuri, "On a Class of Error-Correcting Binary Group Codes," *Information and Control* **3,** 68 (1960.
11. A. Hocquenghem, "Codes Correcteurs d'Erreurs," *Chiffres* **2,** 147 (1959).
12. R. L. Griffith, "Data Recovery in the IBM Photo-Digital Mass Storage System," to be published.


*Received June 27, 1968.*