Management Adviser

Volume 8 | Number 5 Article 4

9-1971

Application and Reliability of the Self-Checking Digit Technique

John O. Mason Jr.

William E. Connelly

In today's environment, traditional controls must often be replaced by electronic controls. The authors discuss the four types of EDP internal control, with particular attention to —

THE APPLICATION AND RELIABILITY OF THE SELF-CHECKING DIGIT TECHNIQUE

by John O. Mason, Jr.
University of Alabama
and William E. Connelly
Touche Ross & Co.

In the development of computer-based information systems, system designers attach considerable weight to internal control features. The emphasis on internal control is related to the effectiveness of information systems. According to Felix Kaufman, "Control is a preeminent condition to data processing effectiveness. A properly controlled system will operate effectively with less than optimal design and equipment. The converse is not true."[1] The point is that computer-based information systems that do not include an adequate plan for control do not function effectively.

In order to add to the system designer's working knowledge of the overall control mechanism of an information system, the authors discuss the usefulness of the self-checking digit technique, an automatic control feature commonly found in information systems. This article describes and illustrates the application of the self-checking digit technique at various control points within an information system, explores advantages and cost of its adoption, presents the results of a simulation experiment conducted by the authors to test the reliability of selected self-checking digit methods, and interprets the findings in terms of which methods are superior in detecting different types of coding errors.

Because of the introduction of computers in business data processing, many traditional control measures are no longer available. However, new methods of control, often referred to as EDP controls, have been devised to substitute for human controls. There are four types of EDP internal control techniques, which may be classified according to control points within the information system:

1. *Source data controls* — which

[1] Kaufman, Felix, *Problems of Control in Electronic Data Processing,* Lybrand, Ross Bros. & Montgomery, New York, 1963, p. 9.

provide control over the creation and handling of source data outside the computer area. These include, among others, predetermined batch control totals, self-checking digit tests, review of source documents for completeness, key-verification, and visual verification.

2. *Hardware controls* — those built into computer hardware by the manufacturer. Some of the control techniques of this type include parity checks, echo checks, dual-gap heads, dual arithmetic circuitry, and sequential arithmetic circuitry.

3. *Program (software) controls*— tests of (a) input fed into the computer configuration to obtain assurance that all transactions transmitted from the recording point have been received at the processing point, and (b) items processed by the computer to determine whether the functioning of computer processing operations is as planned. Some of the control techniques in this area include limit checks, structural checks, alphanumeric checks, internal header and trailer labels, completeness checks, valid field tests, self-checking digit tests, record counts, batch control totals, sequence checking, cross-footing balance checks, and zero-balancing.

4. *Operations controls* — procedural controls over data processing operations within the computer area. Some of the control techniques of this type include the grandfather - father - son technique, remote storage copy of one file generation together with subsequent transactions, file protection ring, documentation, manual reconciliation of batch control totals and record counts, and external file labels.

Controls vary with systems

Not every control feature listed in the above four areas would be used in a given information system. In designing such a system, the system designer should consider the entire set of controls in relation to the nature of the business and the environment in which they are applied, rather than view individual controls in isolation. However, whenever a business uses numeric codes to identify customers, inventory, products, or employees in order to facilitate the processing of transactions against master files and the effect of an incorrect coding is critical, then the self-checking digit technique should be applied.

Dramatic growth in the number of computer installations during the last two decades has given rise to the increasing use of codes in recording and classifying data. Codes are essential in computerized data processing systems for identifying accounts, customers, products, employees, and cost centers because the shorter the identifier of a piece of data, the less costly the processing. With the use of codes, data accepted for analysis can be easily assigned to appropriate accounts, files, reports, and analyses according to desired management information groupings. It is obviously less costly to record and process the number 8 1 4 9 7 3 to identify a customer in a sales transaction, for example, than to use the full customer name. Less key punch operator time is required, less space is needed on transaction records, and less computer operating time is used.

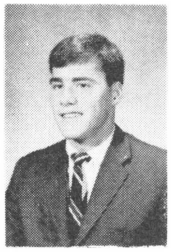
Whereas codes facilitate the classification of data in computer-based information systems, errors in codes give rise to the misclassification of data. Assume that the following transaction is recorded: John Q. Smith purchases ten widgets on account, total price $15.90 (including tax). Under a manual accounting system, the customer would be identified by name on the sales invoice and in the sales journal. Under computerized accounting, Mr. Smith would be identified on a transaction medium (punched card, punched paper tape, magnetic tape) by a code number, say 8 1 4 9 7 3. Furthermore, in the updating of the accounts receivable master file, the number 8 1 4 9 7 3 would be used in matching the transaction against Mr. Smith's master record in the accounts receivable file.

Assume further that in the manual system the source document is improperly completed—that, instead of John Q. Smith, the name John Q. Smyth is recorded. Since the name John Q. Smyth probably is not listed as a customer in the accounts receivable subsidiary ledger, the transaction would not be accepted for updating accounts receivable until the clerk has corrected the error. Assume, however, that in the computer-based system the customer's number is inadvertently miscoded—that, instead of John Q. Smith with account number 8 1 4 9 7 3, Henry J. Green's account number 8 4 1 9 7 3 (transposition error) is introduced at some point in the data stream. Since account number 8 4 1 9 7 3 is also listed in the accounts receivable master file, the transaction would be processed against the wrong master record in the receivables file because the code, though valid, is incorrect.

To eliminate coding errors caused by human failures and machine malfunctions, the system designer must consider incorporating the self-checking digit technique within the overall internal control mechanism of the information system. The self-checking digit tech-

---

JOHN O. MASON, Jr., CPA, is an assistant professor of accounting at the University of Alabama. Previously he was an instructor at the University of Missouri and a staff accountant with a major accounting firm. Dr. Mason received his B.S. and M.S. degrees from Louisiana State University and his Ph.D. from the University of Missouri. He is a member of the American Accounting Association's committee on responsibility for financial statements.



WILLIAM E. CONNELLY, CPA, is a staff accountant with Touche Ross & Co. in Nashville, Tenn. He received his B.S. from David Lipscomb College, in Nashville, and his M.A. from the University of Alabama. Mr. Connelly is a member of the National Association of Accountants and Beta Alpha Psi, the professional accounting fraternity.

nique tests the validity of a numeric code, such as a customer account number in the example above. This is done by applying a mathematical formula to the number itself. The principle behind the technique is that if the formula produces a result equal to the extreme right-hand digit[2] of the number, the number is accepted as valid. If the formula does not produce a result equal to the right-hand digit, the assumption is that an error has occurred in either recording or processing the number. That is, the identification number is not the same at this point in the data stream as it was when it was initially issued to identify a particular data item.

Transposition errors

As a framework in which to illustrate the application of the self-checking digit technique at various control points in a computer-based information system, let us look at the transaction discussed earlier (the sale of widgets to customer John Q. Smith) in which the customer number 8 1 4 9 7 3 was transposed to 8 4 1 9 7 3. Chances are (as will be shown in the next section of this article) the transposition error will cause the transaction to fail the self-checking digit test. Consequently, either the transaction entry will not be transmitted to the computer for processing or, if accepted by the computer, an error message will be printed and the transaction skipped. Only those transactions that pass the self-checking digit test will be processed against the master file.

The self-checking digit technique is not a single technique, but a family of techniques. The four basic methods described below are commonly found in information systems:

1. The Modulus 10 'Simple Sum' method is the easiest to understand because it requires only a few relatively simple calculations. With this method, a self-checking number is formed in the following manner:
   a. Begin with a basic code number
      8 1 4 9 7 3
   b. Sum the digits in the number
      $8 + 1 + 4 + 9 + 7 + 3 = 32$
   c. Subtract the sum from the next highest multiple of 10
      $40 - 32 = 8$
   d. Check digit
      8
   e. Self-checking number
      8 1 4 9 7 3 8

2. The Modulus 10 '2-1-2' method is more complex in that each digit of the basic code number, beginning with the units digit, is weighted by the consecutive factors of 2, 1, 2, 1, 2, . . . In this method, a self-checking number is formed as follows:
   a. Begin with a basic code number
      8 1 4 9 7 3
   b. Apply consecutive weights of 2, 1, 2, 1, 2, . . . to each digit of the basic code number, beginning with the units digit and progressing toward the highest-order digit
      8 1 4 9 7 3
      $\times$ 1 2 1 2 1 2
      $\overline{8}\ \overline{2}\ \overline{4}\ \overline{18}\ \overline{7}\ \overline{6}$
   c. Sum the weighted digits
      $8 + 2 + 4 + 18 + 7 + 6 = 45$
   d. Subtract the sum from the next highest multiple of 10
      $50 - 45 = 5$
   e. Check digit
      5
   f. Self-checking number
      8 1 4 9 7 3 5

3. The Modulus 11 'Arithmetic' method, like the previous method, is based on a weighted scheme, but each digit in the basic code number is weighted by a separate factor.
   a. Begin with a basic code number
      8 1 4 9 7 3
   b. Apply consecutive weights of 2, 3, 4, 5, 6, 7, 2, 3, 4, 5, . . . to each digit of the basic code number beginning with the units digit and progressing toward the high-order digit
      8 1 4 9 7 3
      $\times$ 7 6 5 4 3 2
      $\overline{56}\ \overline{6}\ \overline{20}\ \overline{36}\ \overline{21}\ \overline{6}$
   c. Sum the weighted digits
      $56 + 6 + 20 + 36 + 21 + 6 = 145$
   d. Divide the sum by 11
      $145 \div 11 = 13$ with 2 remaining
   e. Subtract the remainder, 2, from 11
      $11 - 2 = 9$
   f. Check digit (When the arithmetic process generates a result of eleven, the digit 0 is substituted.)
      9
   g. Self-checking number
      8 1 4 9 7 3 9

4. The Modulus 11 'Geometric' method is almost identical to the Modulus 11 'Arithmetic' method, except that the weighting factors are based on a geometric sequence of 2.

With the Modulus 11 'Geometric' method, a self-checking number would be obtained in the following manner:
   a. Begin with a basic code number
      8 1 4 9 7 3
   b. Apply consecutive weights of 2, 4, 6, 8, 16, 32, 64, . . ., 2^n to each digit of the basic code number, beginning with the extreme right-hand digit and progressing toward the high-order digit.
      8 1 4 9 7 3
      $\times$ 64 32 16 8 4 2
      $\overline{512}\ \overline{32}\ \overline{64}\ \overline{72}\ \overline{28}\ \overline{6}$
   c. Sum the weighted digits
      $512 + 32 + 64 + 72 + 28 + 6 = 714$
   d. Divide the sum by 11
      $714 \div 11 = 64$ with 10 remaining
   e. Subtract the remainder, 10, from 11
      $11 - 10 = 1$
   f. Check digit (When the arith-

[2] Although the check digit must appear on the right of the number if the checking is performed on a card punch, it is also quite common to find check digits preceding numbers in systems which use optical readers. Good examples of this are the various credit card accounts used by oil companies, department stores, etc.

metic process generates a result of eleven, the digit 0 is substituted.)
1

g. Self-checking number
8 1 4 9 7 3 1

These four methods of the self-checking digit technique are referred to, respectively, as:

1. Mod 10 Simple Sum
2. Mod 10 Alternate
3. Mod 11 Arithmetic
4. Mod 11 Geometric

The control points at which the self-checking digit methods may be applied to detect and correct errors are: (1) creation and handling of source data outside the computer area and (2) computer processing. As a source data control, the technique provides a means of verifying the accuracy of coded data at the same time it is converted to machine-usable form. The requirements for using the technique at the data conversion station are as follows:

1. The self-checking number feature must be installed on the input preparation device, whether it be a card punch, paper tape punch, or magnetic tape recorder.[3]

2. A check digit must be generated for each basic code number to be self-checked.

For example, the self-checking feature may be installed on an IBM Model 29A Card Punch. The operator, who controls the feature by a toggle switch and special punches in the program card, keys the code number as it appears in the source document. Internal calculations by special circuitry attached to the card punch verify both the accuracy of the keying operation and the validity of the self-checking number as it appears on the source document. When the number on the source document is correct and the number is keyed correctly, the keying operation continues uninterrupted. On the other hand, if the number is not keyed correctly or if the self-checking number appears incorrectly on the source document, the card punch (once the number is keyed by the operator) will signal an error and lock up.[4]

How error is caught

Let us expand the framework established in the preceding section (the sale of widgets to customer John Q. Smith) by adding the assumption that management adopted the designer's recommendation that the Mod 11 Arithmetic check digit method be applied as a source data control. Moreover, assume that John Q. Smith's previous customer number, 8 1 4 9 7 3, was converted to the following self-checking number 8 1 4 9 7 3 9. (The check digit 9 was appended to basic account number 8 1 4 9 7 3 in order to form the self-checking number 8 1 4 9 7 3 9.) Assume further that the number was correctly entered on the sales invoice, but was incorrectly punched by the key punch operator as 8 4 1 9 7 3 9 (transposition error). At the time the number was incorrectly keyed by the operator, the self-checking digit circuitry of the card punch would perform the following internal calculations:

1. Begin with the transposed code number
   8 4 1 9 7 3 9
2. Apply consecutive weights of 2, 3, 4, 5, 6, 7 to each digit of the basic code number
   $$\begin{array}{rrrrrrr} 8 & 4 & 1 & 9 & 7 & 3 & \\ \times\ 7 & 6 & 5 & 4 & 3 & 2 & \\ \overline{56} & \overline{24} & \overline{5} & \overline{36} & \overline{21} & \overline{6} & \end{array}$$
3. Sum the weighted digits
   $56 + 24 + 5 + 36 + 21 + 6 = 148$
4. Divide the sum by 11
   $148 \div 11 = 33$ with 5 remaining
5. Subtract the remainder, 5, from 11
   $11 - 5 = 6$
6. Result
   6
7. Compare result with check digit of code number
   $6 \neq 9$

Since the result calculated by the self-checking digit circuitry does not equal the check digit (step 7 above), a red light will appear on the keyboard and the keyboard will lock. The operator must release the punched card by pressing an error reset key and then repeat the keying process. If the number is correctly punched the second time, the keyboard will not lock and the operator will be able to complete preparation of the transaction card.

Assume, however, that the number was entered incorrectly on the sales invoice. An error will be indicated each time a card is punched from the sales invoice. The second time an error is indicated, the operator will assume that the number on the sales invoice is invalid, a situation which must be remedied before the transaction can be converted to machine-usable form and transmitted to the computer for processing.

The self-checking digit technique also may be incorporated as a part

[3] Installation of the self-checking number feature on the input preparation device was the first use of the check digit technique, but in today's third generation computer world this is not always necessary. Some system designers would prefer that the key-verification technique be used to catch key punch errors and that a complete edit run by the computer, prior to the time transactions are processed against the master file, be used to detect source document coding errors.

[4] *Reference Manual—IBM 29 Card Punch*, IBM, Poughkeepsie, New York, 7th ed., 1970, pp. 28-32.

FIGURE 1

ACCOUNTS RECEIVABLE UPDATING RUN
SEQUENTIAL (BATCH) PROCESSING

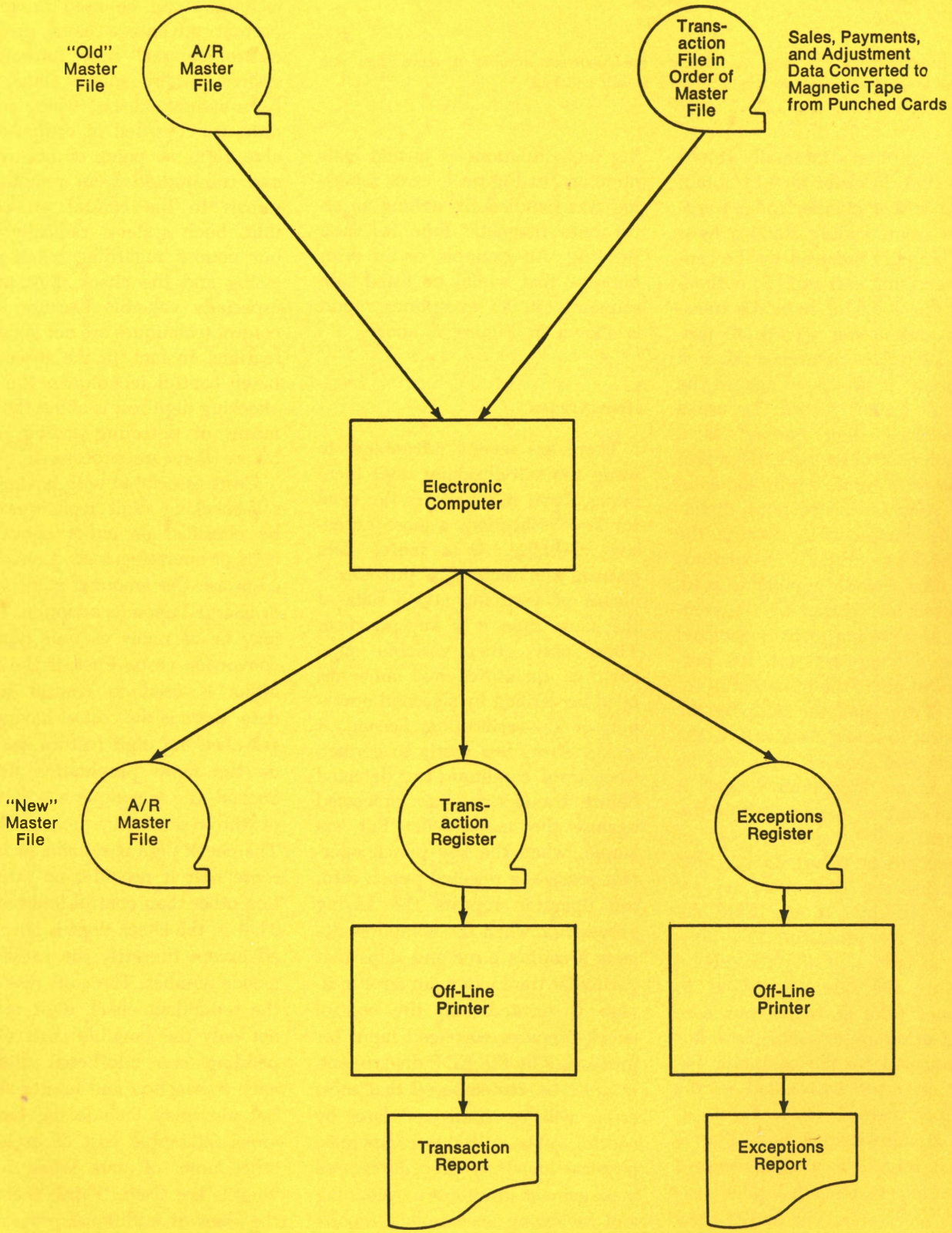


FIGURE 2

EXCEPTIONS REPORT
ACCOUNTS RECEIVABLE UPDATING RUN
FRIDAY, FEBRUARY 19, 1971

CUSTOMER NUMBER	SALES INVOICE	AMOUNT
8419730*	983756	$15.90
etc.	etc.	etc.

* An asterisk would be used to indicate the incorrect number if more than one number in a transaction or master record is self-checked.

of the computer's internally stored instructions in order to: (1) obtain assurance that number codes transmitted from reading stations have been properly received by the central processing unit and (2) authenticate the codes of both the transaction and master records to provide adequate assurance that a transaction is processed against the intended master record. To use a self-checking digit method as a program control requires that a programer incorporate within the computer program instructions directing the computer to execute the self-checking digit calculations each time a code number is read, processed, or written by the computer. If a coding error is detected during a computer run, the program will abort the transaction. Instead of the operator stopping the computer to make a correction, the location and type of error will be listed in an exceptions report. A control clerk or the originating department will be given a copy of the exceptions report to see that the items are corrected and returned promptly to the EDP department for processing. To illustrate, assume that the transaction involving the sale of widgets to customer John Q. Smith was converted to computer-usable form but was rejected by the computer because an error was noted in the customer number code. Figure 1, page 31, illustrates a typical error routine found in a computerized accounting system to update accounts receivable. The output from the updating run ordinarily would include an exceptions report noting such situations as invalid code numbers. In Figure 1, error reporting was handled by writing an error onto magnetic tape for later printing. An example of an error message that would be listed subsequently in an exceptions report is shown in Figure 2, above.

Advantages

There are several advantages to using the self-checking digit technique. First, it eliminates the need for key-verification, a more expensive checking. As a source data control, the technique provides a means of verifying coded data at the same time it is key punched. Thus, only other variable data, such as quantities and amounts, need be verified by a second operation of key-verification. Second, it makes errors less costly to correct, since error conditions are detected before transactions are processed against the master file. For example, when the key punch operator releases a rejected punch card, the operator repeats the keying process; or when the computer detects a coding error and skips that particular transaction, an error message is printed and the control check prepares corrected input for processing by the EDP department. It must be remembered that most errors will be discovered later by control totals, customer complaints, physical inventory procedures, and management intuition in examining and reviewing accountants' reports and analyses. However, the cost of correcting errors can be high if the computer has processed them as if they were correct. The self-checking digit technique obviates the cost of correcting certain file errors, for such errors are detected and corrected before a transaction is processed against the master file.

Third, the self-checking digit technique can be used to an important advantage as a program control in real time information systems where source data have been automated, i.e., where source data are recorded in computer-usable form at point of occurrence and transmitted from remote terminals to the central processing unit. Such systems radically alter our notions regarding batch processing and the check digit test is especially valuable because batch control techniques are not generally feasible. In fact, in the absence of batch control techniques, the self-checking digit test is about the only means of detecting coding errors before they are processed.

Costs associated with having this self-checking digit technique may be classified as either conversion costs or operating costs. Conversion costs are the amounts paid, given, or charged upon its adoption. There may be as many as four types of conversion costs. First, if the technique is used to control source data, there is the cost of having the self-checking digit feature installed on the input preparation device. Second, the technique is a member of the redundancy check family. The check digit is redundant in the sense that it provides no information other than control information. That is, the check digit is not needed except to verify the validity of a code number. The costs related to the redundant check digit include not only the tangible costs of appending one additional digit to each transaction and master record but also may include the less obvious intangible cost of excluding other units of data when record lengths are fixed. Third, there are the costs of additional programing time. Programers must write programs that convert code numbers

to self-checking digit numbers. Moreover, if the technique is implemented as a program control, programers will have to modify existing programs in order to incorporate within them the self-checking digit algorithm (systematic set of equations representing the self-checking digit calculations). Fourth, existing plates or cards encoded with customer, product, employee, or process identification numbers will have to be scrapped and new ones issued.

Operating costs involve the costs of computer time used in performing the self-checking digit algorithm.[5] While the cost of computer time for a single transaction or even one day's transactions may not be material, this will not be true of the costs of additional computer time requirements for an entire year. For example, assume that in the accounts receivable updating run illustrated in Figure 1, the computer is processing 1,000 transactions against a master file of 25,500 records. Even if the self-checking digit algorithm was performed only once a day, six days a week, for each transaction and master record, the calculations would be executed by the computer 8,268,000 different times a year. Even for a computer whose operating cycle is measured in terms of a few hundred nanoseconds (billionths of a second) the total time required for carrying out the self-checking digit calculations would be substantial.

Reliability of methods

How reliable are the self-checking digit methods? The answer to this question was the final objective of the study. This aspect of the study took the form of a simulation experiment, in which the four self-checking digit methods described previously were used in an attempt to detect simulated errors in a hypothetical set of ten thousand seven-digit, self-checking code numbers. In all, attempts were made to detect five different types of errors. They are described below:

Assume the correct number is
8 1 4 9 7 3 9

1. Single transcription error — where one digit is copied or processed incorrectly
   8 3 4 9 7 3 9
2. Single transposition error — where the position of two digits in a number is interchanged
   8 4 1 9 7 3 9
3. Double transposition error — where the position of two sets of digits in a number is interchanged
   8 4 1 7 9 3 9
4. Random scramble — where the entire number is garbled
   7 6 2 3 5 9 6
5. Substitution of a valid, but incorrect number
   6 5 2 0 1 0 3

Before presenting the results of the simulation, a few comments about the methodology are in order. First, for each check digit method, ten thousand self-checking numbers were generated. Second, an error of each type illustrated above was simulated in each of the self-checking numbers; and an attempt was made to detect such errors by means of the self-checking digit algorithm. Third, the percentage of errors detected was computed by type of error. Fourth, this procedure was repeated for each of the four check digit methods; the results appear in Table 1, page 34. The percentage of detected errors may be taken as an index of reliability—the greater the percentage of errors detected, the higher the degree of reliability present.

Clearly, all methods are not equally reliable. The ability to detect errors is greatest in the Mod 11 methods. In all error categories the Mod 11 methods detected coding errors as well as or better than

There are several advantages to using the self-checking digit technique. First, it eliminates the need for key-verification, a more expensive checking. As a source data control, the technique provides a means of verifying coded data at the same time it is key punched.

[5] In batch processing operations, additional computer time will not be required to perform the self-checking number calculation if this edit technique is carried out when punched cards are loaded to magnetic tapes and edit processing operations are overlapped with input/output operations.

TABLE I

RELIABILITY FACTORS ASSOCIATED WITH SELF-CHECKING DIGIT METHODS (ROUNDED TO THE NEAREST PER CENT)

SELF-CHECKING DIGIT METHOD / TYPE OF ERROR	SINGLE TRANSCRIPTION	SINGLE TRANSPOSITION	DOUBLE TRANSPOSITION	RANDOM SCRAMBLE	SUBSTITUTION OF VALID, BUT INCORRECT NUMBER
Mod 10 — Simple Sum	100%	0%	0%	90%	0%
Mod 10 — Alternate	94%	90%	90%	90%	0%
Mod 11 — Arithmetic	100%	90%	90%	90%	0%
Mod 11 — Geometric	100%	90%	90%	90%	0%

the Mod 10 methods. There is an extremely small probability that these results were due to chance (less than one in a thousand).

With the exception of single transcription errors, the Mod 10 Alternate method was third best. The Mod 10 Simple Sum method performed least well (except in the single transcription error category). The most crucial factor affecting reliability seemed to be the weighting of digits (versus non-weighting), while the weighting scheme was a somewhat less important factor.

None of the methods is 100 per cent reliable. In fact, all are quite powerless to combat a special type of error—that of assignment of an incorrect, but valid code number for another. The explanation of reliability is an interesting topic, but is not explored here because of an already lengthy article. However, part of the answer can be traced to the fact that self-checking digit methods are capable of generating check digits which have but ten possible values, 0 through 9. Because more than one code number will be assigned the same check digit, there is always the possibility that a substitution or transposition of digits in one number may result in another valid number (the latter number having the same check digit as the former number).

Summary

Nearly two decades have passed since the first commercially available computer was introduced in the United States. Since that time, as Geoffrey Horwitz recently pointed out, the number of computers installed within the United States has doubled every three years and this rate of increase is expected to continue.[6] As the number of installations continues to increase, the system designer's emphasis on problems of achieving proper control of computer-based information systems will probably persist.

In this article, the writers have illustrated the application of an automatic control feature at various control points within an information system. Designated the self-checking digit technique, it tests the accuracy of coded data during: (1) conversion to machine-usable form and (2) computer processing. Because the technique makes it possible for an information system to detect incorrectly coded transactions before they are assigned to accounts, files, reports, and analyses, errors are less costly to correct. Another cost advantage associated with the application of this control feature is that it partially eliminates the need for the more costly key-verifying operation.

The writers also investigated the reliability of four check digit methods found in information systems. The results of a simulation experiment indicated that, contrary to claims of upwards of 100 per cent effectiveness, none of the methods is nearly that reliable. Moreover, they vary substantially in ability to detect coding errors; in all error categories considered, the Mod 11 methods detected coding errors as well as or better than the Mod 10 methods.

Thus, the self-checking digit technique provides information systems with an automatic control device for detecting coding errors. Though not 100 per cent effective, when complemented by predetermined batch total techniques, the check digit test greatly strengthens internal control over EDP operations.

[6] Horwitz, Geoffrey B., "EDP Auditing —The Coming of Age," *Journal of Accountancy*, American Institute of CPAs, New York, August, 1970, p. 48.