where

$$c_1(i) \triangleq \sum_{n=0}^{N-i-1} PN_n PN_{n+i} \tag{A8}$$

$$\hat{c}_1(i) \triangleq \sum_{n=N-i}^{N-1} PN_n PN_{n+i-N} \tag{A9}$$

$$\Delta_1(i) \triangleq c_1(i+1) - c_1(i) \tag{A10}$$

$$\hat{\Delta}_1(i) \triangleq \hat{c}_1(i+1) - \hat{c}_1(i) \tag{A11}$$

and where PN_j is the jth chip of the PN sequence and T_c is the chip duration.

$$K_2 = \frac{1}{T}\sum_{i=0}^{N-1}\left\{A(i)T_c + \left(B(i) - \frac{A(i)}{T}\right)\frac{(i+1)^2T_c^2 - i^2T_c^2}{2} + \left(C(i)\frac{B(i)}{T}\right)\frac{(i+1)^3T_c^3 - i^3T_c^3}{3} - \frac{C(i)}{T}\frac{(i+1)^4T_c^4 - i^4T_c^4}{4}\right\} \tag{A12}$$

where

$$A(i) \triangleq c_1(i)\hat{c}_1(i)T_c^2 - iT_c^2(\hat{c}_1(i)\Delta_1(i) + \hat{\Delta}_1(i)c_1(i)) + i^2T_c^2\Delta_1(i)\hat{\Delta}_1(i) \tag{A13}$$

$$B(i) \triangleq T_c(\hat{c}_1(i)\Delta_1(i) + c_1(i)\hat{\Delta}_1(i)) - 2T_c i\Delta_1(i)\hat{\Delta}_1(i) \tag{A14}$$

and

$$C(i) \triangleq \Delta_1(i)\hat{\Delta}_1(i). \tag{A15}$$

ACKNOWLEDGMENT

The authors would like to gratefully acknowledge the help of D. Van Renen for carefully reviewing the details of the analysis, and M. Mammone and Dr. F. Hemmati for generating all the numerical results.

REFERENCES

[1] D. L. Schilling, L. B. Milstein, R. L. Pickholtz, and R Brown, "Optimization of the processing gain of an M-ary direct sequence spread spectrum communication system," *IEEE Trans. Commun.*, vol. COM-28 pp. 1389–1398, Aug. 1980.

[2] L. B. Milstein, S. Davidovici, and D. L. Schilling, "The effect of a comb jammer on a direct sequence spread spectrum communication system," in *Proc. 1980 Nat. Telecommun. Conf.*, pp. 69.8.1–69.8.5.

[3] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. New York: Wiley, 1965, pp. 97–101.

[4] D. E. Borth and M. B. Pursley, "Analysis of direct-sequence spread spectrum multiple-access communications over Rician fading channels," *IEEE Trans. Commun.*, vol. COM-27, pp. 1566–1577, Oct. 1979.

[5] M. Schwartz, W. R. Bennett, and S. Stein, *Communications Systems and Techniques*. New York: McGraw-Hill, 1966, ch. 9.

[6] R. S. Kennedy, *Fading Dispersive Communication Channels*. New York: Wiley, 1969.

[7] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*. Nat. Bur. Stand., 1964, p. 377.

An Arithmetic Checksum for Serial Transmissions

JOHN G. FLETCHER

***Abstract*—An error-detection method for serial transmissions is presented that uses an integer arithmetic checksum. The checksum is compared with cyclic-redundancy-check methods with regard to error-detecting ability and efficiency of implementation in software (or firmware). The method is a bit weaker at detection but is more efficient, thus representing a different and potentially useful choice in the cost-benefit spectrum.**

I. INTRODUCTION

Serial transmission of information between computers is subject to errors that arise from environmental disturbances and hardware malfunction. It is common practice to append to each transmission a number of redundant check bits in order to detect these errors. The sender computes these bits as a function of the preceding bits in the transmission; the receiver performs the same computation and accepts the transmission as error-free only if the received check bits are the same as those computed.

The generation and verification of check bits is frequently carried out by special hardware that is part of the transmission equipment. In these situations, the check bits are usually computed according to a cyclic-redundancy-check (CRC) algorithm. In such an algorithm, the bits of a transmission are treated as the successive coefficients of a polynomial over the binary field (the field of integers modulo 2), and the check bits are chosen so that this polynomial is exactly divisible by a preselected polynomial that is a parameter of the algorithm. More complete explanation of CRC's may be found elsewhere [1]. The wide popularity of CRC's stems primarily from two facts: 1) they have several proven, desirable properties with regard to the kind and number of errors they will detect; and 2) they are conveniently and efficiently implemented by hardware.

However, on most computers CRC's are not conveniently and efficiently implemented by software (or firmware). This is because most computers do not have instructions oriented toward computation involving polynomials over a binary field; rather, their instructions are oriented toward conventional integer arithmetic. Programmers of CRC procedures must choose between iterative loops that shift and test each bit of a transmission or methods that treat short blocks of bits by table look-up [2]. These procedures are significantly less efficient with respect to execution time (looping) or storage requirements (tables) or both than they would be if integer arithmetic could be used. In spite of this, CRC procedures are often implemented in software (or firmware). This

Paper approved by the Editor for Data Communication Systems of the IEEE Communications Society for publication without oral presentation. Manuscript received June 8, 1979; revised August 6, 1981. This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

The author is with the Lawrence Livermore Laboratory, University of California, Livermore, CA 94550.

0090-6778/82/0100-0247$00.75

may be necessary because the transmission equipment at the other end of the communication channel uses CRC hardware. However, when both ends of a channel implement the redundancy check in software (or firmware) (as in the fairly common case of two mini-, or micro-, computers communicating over a simple, one-byte-at-a-time, asynchronous channel), it would seem that a redundancy check based on integer arithmetic would be preferable because of increased speed and reduced storage requirements. This assumes that the arithmetic redundancy check (checksum) has properties as desirable, or nearly as desirable, as a CRC.

The purpose of this paper to describe an arithmetic checksum for serial communication and to compare its error-detecting properties to those of CRC's. As will be seen, the properties of the checksum are not quite as good as those of a CRC. The choice between the two is, therefore, a tradeoff between degree of efficiency and degree of error-detection.

II. AN ARITHMETIC CHECKSUM

A transmission is treated as a sequence of K-bit bytes. The design of most communication interfaces indicates the choice $K = 8$. Each byte is treated as an integer, and arithmetic is performed on these integers modulo M. The only values of M considered in detail are $M = 2^K$ (two's complement arithmetic) and $M = 2^K - 1$ (one's complement arithmetic). In the former case, if an addition overflows beyond K bits, the overflow is discarded. In the latter case the overflow is added back into the result as an "end-around" carry. Either case is efficiently implemented on most computers. Even though two's complement is usually the more efficient of the two, it is shown below that there are reasons to prefer one's complement.

The last R bytes of a transmission (assumed to be a whole number of bytes) are the check bytes. The case $R = 2$ (when $K = 8$) is most important, because it results in the same number of check bits, $C = KR = 16$, as a conventional CRC. The sender and receiver each maintain a record consisting of R one-byte checksums $C(r)$, $r = 0, 1, \cdots, R - 1$, that are all initialized to zero at the beginning of a transmission. As each byte B is sent or received, it is incorporated into the checksums as follows:

$$\begin{aligned} &C(0) \leftarrow C(0) + B; \\ &C(1) \leftarrow C(1) + C(0); \\ &\quad \cdots \\ &C(r) \leftarrow C(r) + C(r-1); \\ &\quad \cdots \\ &C(R-1) \leftarrow C(R-1) + C(R-2). \end{aligned} \tag{1}$$

So only R additions per byte transmitted are needed to compute the checksums.

When a transmission is complete, the receiver expects each $C(r)$ to equal zero. The sender arranges this (unless there is a transmission error) by choosing each of the R check bytes as follows:

$$B \leftarrow -\sum_{r=0}^{R-1} C(r). \tag{2}$$

Each check byte B is computed by using the then current values of the $C(r)$. The byte is then included into the checksums according to (1), thus modifying the $C(r)$ so that the computation of the next check byte will, in general, yield a different value. It is easy to see from (1) and (2) that $C(R-1)$ is zero after the first check byte is included in the checksums. Then $C(R-2)$, as well as $C(R-1)$, is zero after the second check byte is included, and so on until all $C(r)$ are zero (as desired) after the last (Rth) check byte is included. This also means that the lth check byte may be computed by changing the upper limit in the sum in (2) to $R - l$, because the omitted terms are all known to be zero. In particular, the last check byte (which is also the last byte of the transmission) is chosen to be simply $-C(0)$.

Speedy and compact algorithms embodying (1) and (2) are easily implemented on typical modern computers, as should be clear. When implementing, it may be convenient to change the connecting sign on the right-hand side of one or more of the statements of (1) from plus to minus. Doing so has the effect of replacing some $C(r)$ by their negatives, and compensating changes of sign must be made in (2). Since all $C(r)$ ultimately become zero, the check bytes themselves are unchanged (except for possibly switching from one to the other of the two one's complement zeros). Therefore, this freedom of connecting-sign selection may be independently exercised by the sender and the receiver.

III. ANALYSIS OF THE CHECKSUM

The remaining issue is—how good is the checksum at error detection? The analysis begins by examining the consequences of (1). If $B(1), B(2), \cdots, B(N)$ are the bytes in the order transmitted, then the value of $C(r)$ after L bytes have been transmitted is

$$C(r, L) = \sum_{l=0}^{L-1} \frac{(l+r)!}{l!r!} B(L-l). \tag{3}$$

This is readily proven by induction on both r and L, starting with $C(r, 0) = 0$ and the rather obvious fact that $C(0, L)$ is simply the sum of $B(1)$ through $B(L)$. Primary interest centers on the case $L = N$, the total number of bytes in the transmission. Then the $C(r, N)$, as computed by the sender, using the bytes sent as the $B(N-l)$, equal zero for all $r < R$. The $C(r, N)$ as computed by the receiver are, therefore, given either by using the bytes received as the $B(N-l)$ or instead by using the differences between the received and sent bytes, that is, the numerical values of the errors in the received bytes. The latter choice is made hereafter, and (3) is viewed as expressing $C(r, L)$ in terms of the errors $B(L-l)$ in the received bytes. An error is detected if and only if $C(r, N) \neq 0$ for at least one $r < R$.

Consider now the following, which is similar but not identical to (3):

$$E(r, L) = \sum_{l=-\infty}^{\infty} \frac{(l+r)!}{l!r!} B(L-l). \tag{4}$$

The convention is made that $B(L-l) = 0$ if $(L-l) \leqslant 0$ or $(L-l) > N$; that is, the transmission is treated as being preceded and followed by infinite sequences of error-free bytes. Also, the coefficient $(l+r)!/l!r!$ is defined as $(l+1)(l+2)\cdots(l+r)/r!$ for all positive, zero, and negative integers l, so

long as r is a nonnegative integer. It is easy to see that $E(r, L)$ and $C(r, L)$ are, in general, different quantities except for the case $L = N$, when $E(r, N) = C(r, N)$. The following lemma holds.

Lemma: An error is detected, that is, there exists an $r < R$ such that $C(r, N) = E(r, N) \neq 0$, if there exists an $r < R$ and an L such that $E(r, L) \neq 0$. No error is detected (which may be because there is in fact no error), that is, for each $r < R$, $C(r, N) = E(r, N) = 0$, if for each $r < R$ there exists an L such that $E(r, L) = 0$.

The lemma is an immediate corollary to the assertion that, if for each $r < R$, $E(r, L) = 0$ for one value of L (which may differ for each r), then $E(r, L) = 0$ for all $r < R$ and L. This assertion is proven by induction on both r and L. For each r, the induction on L starts with that L for which $E(r, L) = 0$ is hypothesized and is two-way so as to extend the result to both larger and smaller L. For $r = 0$, the induction is simply the observation that $E(0, L)$ is independent of L, because all the coefficients in the sum are equal to one. For $r > 0$, the induction uses the facts that $E(r, L + 1) = E(r, L) + E(r - 1, L + 1)$ and $E(r, L - 1) = E(r, L) - E(r - 1, L)$, which follow from the identity $(l + r)!/l!r! = (l + r - 1)!/(l - 1)!r! + (l + r - 1)!/l!(r - 1)!$

The foregoing lemma is the basic element in the subsequent analysis of the checksum algorithm. For each class of errors considered, a single convenient choice of L is made for each $r < R$. To ascertain the fraction of undetected errors, we determine the fraction of errors in the class for which each $E(r, L) = 0$. The following classes of error considered are the same as those usually considered in analyzing a CRC: 1) all possible errors (involving any number of bits); 2) errors confined to a single burst of length C, where $C = KR$ is the number of check bits; 3) errors affecting an odd number of bits; and 4) errors affecting two bits [1].

IV. DISTRIBUTION OF ERRORS

As a preliminary to discussing these classes of errors, let us consider the possible errors in a single K-bit byte. Each of the possible 2^K bit patterns has a numerical value m in the range $0 \leq m < M$; these values have a frequency distribution $f(m)$ such that the value m occurs for a fraction $f(m)$ of the bit patterns. The values of the error B in a byte (difference between the received and sent values) similarly have a distribution expressing their frequency of occurrence among the 2^{2K} bit patterns of received and sent byte pairs. For two's complement arithmetic, each of the $M = 2^K$ values of a byte is represented by one bit pattern; so $f(m) = 1/2^K$ for all m. It is not difficult to see that this same uniform distribution also applies to the byte errors. For one's complement arithmetic, the value $m = 0$ is represented by two bit patterns, while the rest of the $M = 2^K - 1$ values have one representation; so $f(0) = 1/2^{K-1}$ and $f(m) = 1/2^K$, $m \neq 0$. The distribution of byte errors can be computed by considering various cases, namely, whether the sent and received bytes are zero or nonzero and whether the two bytes are equal; it is found that $f(0) = (2^{K-1} + 1)/2^{2K-1}$ and $f(m) = (2^K + 1)/2^{2K}$, $m \neq 0$.

If the received and sent bytes have the same bit pattern, no error has occurred. These nonerrors constitute $1/2^K$ of the "errors" enumerated above and necessarily correspond to the value $m = 0$. Therefore, $f(0) - 1/2^K$ represents the fraction of the errors in a single byte that cannot possibly be detected, because only the numerical values of errors affect the values of the $E(r, L)$. In the two's complement case this fraction is zero, but in the one's complement case it is $1/2^{2K-1}$, which corresponds to the possibility that one of the two representations of zero gets changed into the other. This is a shortcoming of one's complement. However, it is worth noting that such an error involves a change in each of the K bits of a single byte and might, therefore, be deemed relatively unlikely. On the other hand, the fact that all the erroneous bits assume the same value may increase the likelihood. This could be overcome by exclusive or-ing a fixed pattern into each byte before submitting it to the checksum algorithm. On an asynchronous channel, moreover, if many adjacent bits assume the same erroneous value, then damage to the byte framing has probably occurred; this should be detected by the hardware.

With two's complement, where $f(m)$ for a single byte is constant (the distribution is uniform), $f(m)$ is also constant for the difference between two bytes. With one's complement, where the values of $f(m)$ for a single byte extend over a range $1/2^K$ wide, the range of values of $f(m)$ for the difference of two bytes is narrower, only $1/2^{2K}$. These facts illustrate the following useful principle.

Thermodynamic Principle: The distribution of values of a sum is no less (usually more) uniform than that of any of the addends. In particular, if any addend has a uniform distribution, then so too does the sum.

This principle is so named because it expresses a tendency away from diversity that is reminiscent of the second law of thermodynamics. The measure of uniformity used is made explicit in the following proof.

The finite Fourier transform (or characteristic statistical function) of a distribution $f(m)$ is defined as

$$\hat{f}(j) = \sum_{m=0}^{M-1} f(m) e^{-2\pi ijm/M}$$

which can be inverted to give

$$f(m) = (1/M) \sum_{j=0}^{M-1} \hat{f}(j) e^{2\pi ijm/M}.$$

Either of these expressions may be derived from the other by using a well-known property of the complex exponential,

$$\sum_{m=0}^{M-1} e^{2\pi ijm/M} = M, \qquad j = 0 \pmod{M},$$

$$= 0, \qquad j \neq 0 \pmod{M}.$$

This same property may be used to demonstrate the equivalence of two definitions for the norm $\| f \|$ of the distribution,

$$\| f \| = \sum_{m=0}^{M-1} f(m)^2 = (1/M) \sum_{j=0}^{M-1} |\hat{f}(j)|^2.$$

The fact that the $f(m)$ are real and nonnegative and have a sum equal to one implies that

$$\hat{f}(0) = 1, \qquad |\hat{f}(j)| \leq 1.$$

The $\hat{f}(j)$, $j \neq 0$, represent nonuniform (varying) components of $f(m)$; therefore, their magnitudes are measures of nonuniformity. In fact, all $\hat{f}(j)$ equal zero, $j \neq 0$, only for the uniform

distribution $f(m) = 1/M$. A good single measure of nonuniformity is $\|f\|$, which assumes its minimum value of $1/M$ only for the uniform distribution.

If two quantities with distributions $g(m)$ and $h(m)$ are added modulo M, the distribution $f(m)$ of their sum is given by the convolution

$$f(s) = \sum_{m=0}^{M-1} g(m)h(s-m)$$

where $(s - m)$ is computed modulo M. From this it follows that

$$\hat{f}(j) = \hat{g}(j)\hat{h}(j).$$

The fact that neither $|\hat{g}(j)|$ nor $|\hat{h}(j)|$ exceeds one implies that

$$|\hat{f}(j)| \leqslant |\hat{g}(j)|, \quad |\hat{f}(j)| \leqslant |\hat{h}(j)|$$

and

$$\|f\| \leqslant \|g\|, \quad \|f\| \leqslant \|h\|,$$

that is, $f(m)$ is no less uniform than $g(m)$ or $h(m)$. This result is readily generalized by induction to more than two addends, and the thermodynamic principle is proven.

The principle is now applied to the sums in (4) that define the $E(r, L)$. The terms in these sums do not necessarily have the same distribution of values as do the $B(L - l)$, because if the coefficient $(l + r)!/l!r!$ of a term shares a common factor F with M, then the entire term has this factor. This means that $f(m)$ for the term equals zero for values of m not divisible by F; the distribution is "crowded" into the values of m that are divisible by F and, therefore, is usually less (never more) uniform than the distribution of $B(L - l)$. However, if the coefficient is relatively prime to M, no such crowding occurs, and the distribution of the term is as uniform as that of $B(L - l)$. Many, often most, of the coefficients for each value of r are in fact relatively prime to M. The thermodynamic principle therefore implies that the distribution of each $E(r, L)$ is at least as uniform as that of the $B(L - l)$, i.e., uniform for two's complement and within a range of $1/2^{2K}$ for one's complement. Further, in the limit as the length N of the transmission becomes large, the distribution becomes uniform even in the one's complement case.

The first class of errors to be analyzed, all possible errors, is usually considered in the limit as the length of the transmission becomes large. Then, as just noted, each $E(r, L) = 0$ for $1/M$ of the errors. Therefore, for $1/M^R$ of the errors, one $E(r, L)$ equals zero for each $r < R$ (and according to the lemma, an error is not detected). This is true provided that $E(r, L) = 0$ are independent conditions for different r. That they are, in fact, independent can be seen by choosing $L = 0$ for all r. Then the coefficients of $B(1), B(2), \cdots, B(r)$ are 0 and of $B(r + 1)$ is $(-1)^r$. So the condition $E(r, 0) = 0$ can be viewed as determining $B(r + 1)$ as a function of $B(r + 2)$, $B(r + 3)$, $\cdots$, $B(N)$. The condition for each successively larger value of r does not involve those $B(-l)$ determined by the conditions for smaller r and is, therefore, independent of them.

The fraction of all "errors" that are not actually errors is $1/2^{KN}$, which becomes negligible for large N. So $1/M^R$ is the fraction of all errors that are undetected. For two's complement ($M = 2^K$), this equals $1/2^{KR}$, exactly the same as for a CRC using the same number of check bits $C = KR$ [1]. For one's complement ($M = 2^K - 1$) the fraction is only slightly more. In the important case $K = 8, R = 2$, two's complement checksum and CRC fail to detect 0.001526 percent of all errors, while one's complement fails to detect 0.001538 percent. In summary, the following has been proven.

Theorem 1: Except for a higher order effect in the one's complement case, the checksum (like a CRC) detects all but a fraction $1/2^C$ of all errors in the limit of long transmissions, where C is the number of check bits.

V. BURST ERRORS

A burst error is an error in transmission in which the erroneous bits are confined to a single block of consecutive bits no longer than the length of the burst. The just-analyzed class of all possible errors is the special case in which the burst is of the same length KN as the transmission. The next class of errors to be examined includes those bursts of length equal to the number of check bits $C = KR$. In general, such a burst begins somewhere within a byte and ends equally far into the Rth byte following. It is convenient to choose the parameter L in (4) for all r so that the burst includes the last $(K - b)$ bits (where $0 \leqslant b < K$) of the Lth byte, all bits of bytes $L + 1$, $L + 2$, $\cdots$, $L + R - 1$, and the first b bits of the $(L + R)$th byte. Because low-order bits of a byte are conventionally transmitted first, the error $B(L)$ in the Lth byte is a multiple of 2^b, the errors $B(L + 1)$, $B(L + 2)$, $\cdots$, $B(L + R - 1)$ are unrestricted, and the error $B(L + R)$ is less than 2^b in magnitude.

For each $r < R$, the coefficient of $B(L)$ is 1, of $B(L + 1)$, $B(L + 2)$, $\cdots$, $B(L + r)$ is 0, and of $B(L + r + 1)$ is $(-1)^r$. So for $r = R - 1$, the condition $E(R - 1, L) = 0$ becomes $B(L) - (-1)^R B(L + R) = 0$. The restrictions on $B(L)$ and $B(L + R)$ imply that $B(L) - (-1)^R B(L + R)$ has the same distribution of values $f(m)$ as an unrestricted $B(L - l)$. So the value zero occurs (and the condition is satisfied) for $1/2^K$ of the errors (two's complement) or $1/2^{2K-1}$ more than this (one's complement). The condition $E(r, L) = 0$ for each successively smaller value of r can be viewed as determining $B(L + r + 1)$ as a function of $B(L)$ and of $B(L + r + 2)$, $B(L + r + 3)$, $\cdots$, $B(L + R)$, which are already determined by the conditions for larger r. So out of all the errors that satisfy the conditions for larger r, the fraction that also satisfies the condition for r is $1/2^K$ (two's complement) or either $1/2^{2K-1}$ or $1/2^{2K}$ more than this (one's complement, where the two alternatives apply depending on whether the value determined for $B(L + r + 1)$ is or is not zero). So, multiplying these fractions together, one finds that all $E(r, L)$ equal zero, $r < R$ (and according to the lemma, an error is not detected), for $1/2^{KR}$ of the burst errors when two's complement is used. The fraction is higher than this when using one's complement. In the worst case (which is obtained for the $1/K$ of the errors where $b = 0$ and the burst lines up on byte boundaries), it is higher by about $R/2^{KR+K-1}$. No attempt is made here to get a better bound than this when $b \neq 0$.

After the fraction $1/2^{KR}$ of the "errors" that are not actually errors (the received and sent byte sequences being the same) are subtracted out, one finds that for two's complement, as for a CRC [1], no burst errors of length $C = KR$ are undetected. One's complement, on the other hand, fails to detect a fraction that is no more than about $R/2^{KR+K-1}$.

In the important case $K = 8$, $R = 2$, the exact fraction is 0.000019 percent. The fact that one's complement cannot detect all C-bit burst errors is of course evident from the fact that it cannot detect a K-bit burst in which one of the two forms of zero is altered into the other. In summary, the following has been proven.

Theorem 2: The two's complement checksum (like a CRC) detects all burst errors of length C, where C is the number of check bits. The one's complement checksum detects all but a fraction on the order of $C/K2^{C+K-1}$, where K is the number of bits in a byte.

VI. PARITY CHECKING

Many CRC's parity check the entire transmission and, therefore, detect all errors affecting an odd number of bits [1]. There is no corresponding property for the checksum. However, there seems to be no reason to regard this as a deficiency. It has already been shown that the checksum detects essentially the same fraction of all errors as does a CRC; so whatever the checksum lacks in detecting odd-bit errors must be made up by improved detection of even-bit errors. Perhaps (although this is not proven) it detects more double-bit errors.

One case of odd-bit errors is very important, namely, single-bit errors. The checksum, like a CRC, detects them all, because such an error changes some $B(L - l)$, and therefore $E(0, L)$, by $\pm 2^b (0 \leqslant b < K)$. So the following holds.

Theorem 3: The checksum (like a CRC) detects all single-bit errors.

VII. DOUBLE-BIT ERRORS

A great deal of the complexity in the treatment of the preceding analysis has only been necessary to properly treat the one's complement case. All that has been gained so far from this complexity is the discovery that one's complement is slightly worse than two's complement. However, as is about to be shown, one's complement is significantly better than two's complement in regard to the last class of errors to be considered—errors affecting two bits—although neither version of the checksum is as good as a CRC. The aspect of undetected double-bit errors that usually is analyzed is their spacing: a suitable CRC can detect all double-bit errors provided that the two bits are less than $2^C - 1$ bits apart [1].

In order that $E(0, L) = 0$ for a double-bit error, the two erroneous bits must necessarily be in the same bit position in different bytes, one being a change from 0 to 1 and the other a change from 1 to 0; then the error in one byte is $2^b (0 \leqslant b < k)$ and in the other is -2^b. It is convenient to choose the parameter L for each r in (4) so that the second-occurring of the two errors is $B(L + 1)$; then its coefficient is 0 for $r > 0$. Therefore, if the spacing of the erroneous bits is $(l + 1)$ bytes so that the other error is $B(L - l)$, then the error is undetected only if $2^b (l + r)!/l!r! = 0 \pmod M$ for $0 < r < R$.

The weakness of two's complement in regard to this condition stems basically from the fact that M (which equals 2^K) has many factors of 2. In particular, in the important case $R = 2$ (where only $r = 1$ satisfies $0 < r < R$), suppose that $b = K - 1$, that is, the highest order bits of two bytes are erroneous. Then for $l = 1$ and $r = 1$, the value of $2^b (l + r)!/l!r!$ is $2^K = 0 \pmod M$. The error is undetected when the erroneous bits are only $(l + 1) = 2$ bytes apart! Although the situation improves slightly for larger R, it is clear that two's complement is entirely inadequate with regard to the property under consideration and can be used in the checksum algorithm only if one regards the property as unimportant. Further consideration is given only to one's complement, where M (which equals $2^K - 1$) is not divisible by 2. This means that 2^b can be divided out of the condition for nondetectability, which becomes $(l + r)!/l!r! = 0 \pmod M$ for $0 < r < R$.

TABLE I
A COMPARISON OF THE ERROR-DETECTING PROPERTIES OF A TYPICAL CRC, THE ONE'S COMPLEMENT CHECKSUM, AND THE TWO'S COMPLEMENT CHECKSUM WHEN TWO 8-BIT CHECK BYTES ARE USED

	CRC	Ones-Complement	Twos-Complement
Fraction of all errors undetected	.001526%	.001538%	.001526%
Fraction of 16-bit burst errors undetected	none	.000019%	none
Single bit errors undetected	none	none	none
Minimum separation of undetected double bit errors	65535	2040	16

For $r = 1$, this condition becomes $(l + 1) = 0 \pmod M$, and the least value satisfying it is $l = M - 1$. So, for $R \geqslant 2$ the spacing between the erroneous bits must be at least $(l + 1) = M$ bytes or $K(2^K - 1)$ bits. This is smaller than the spacing $2^{KR} - 1$ achieved by a suitable CRC for any $R \geqslant 2$. In the important case $K = 8$, $R = 2$, the spacing is 2040 bits (checksum) versus 65 535 bits (CRC), a ratio of about 32. Nevertheless, the checksum provides a substantial spacing that should be quite adequate in many applications.

For $r = 2$, the nondetectability condition becomes $(l + 1)(l + 2)/2 = 0 \pmod M$, which (like the condition for $r = 1$) is satisfied for $l = M - 1$, because 2 does not divide $(l + 1) = M$. So, having three check bytes ($R = 3$) is no better with regard to spacing than having two. Continuing to larger r (and R) may produce a gradual improvement. For example, in the case $K = 8$, where $M = 255$ is divisible by 3, the $r = 3$ condition forces the spacing up by a factor of 3 to 6120 bits. Full analysis of the matter would require delving further into the theory of numbers than seems worth the effort, because the improvement in spacing does not seem to be great enough to be a good reason for using additional check bytes, and in any event two check bytes is the conventional choice. In summary, the following has been proven.

Theorem 4: The one's complement checksum with at least two check bytes detects all double-bit errors, provided that the erroneous bits are spaced by less than $K(2^K - 1)$ bits, where K is the number of bits in a byte. (This is a smaller spacing than that achieved by a suitable CRC.)

VIII. CONCLUSION

One's complement checksum, while not quite as good at error detection as a suitable CRC with regard to the properties examined, makes a very respectable showing (see summary in Table I). Also, since its computation involves no table look-up and only a few (for 16-bit checksums, two) additions per 8-bit byte transmitted, it is more efficiently implemented in software (or firmware) on typical processors than is a CRC. Overall it represents a very reasonable tradeoff between effectiveness and efficiency, which should make it appropriate for use in many situations. It has been implemented successfully in connection with SCULL, a link-level communication protocol [3] used in the Octopus computer network at the Lawrence Livermore National Laboratory.

REFERENCES

[1] W. Peterson and D. Brown, "Cyclic code for error detection," *Proc. IRE*, vol. 49, p. 228, Jan. 1961; also, W. Peterson, *Error Correcting Codes*. Cambridge, MA: MIT Press, 1961.

[2] S. Wecker, "A table-lookup algorithm for software computation of cyclic redundancy check (CRC)," Digital Equipment Corp., Maynard, MA, 1974.

[3] J. Fletcher, "Serial communication protocol simplifies data transmission and verification," *Comput. Design*, p. 77, July 1978.

Message Error Detecting Properties of HDLC Protocols

G. FUNK

Abstract—**Upgraded requirements for data integrity and data efficiency in real-time process control applications necessitate critical investigations on existing standard data transmission conventions, such as the HDLC (high level data link control) protocol.**

It is shown that a single bit error within an ordinary HDLC frame may cause an undetectable message error at the receiver. Further proposals for using modified HDLC protocols which guarantee the detection of single bit errors fail to detect double bit errors within a frame. Refined assessments for corresponding nondetectable message error probabilities are based on bit sequence probabilities in HDLC frames which differ slightly from bit sequences in original text fields due to the "bit escaping mechanism" required by the HDLC protocol.

SUMMARY

An investigation of the suitability of various standard communication protocols for telecontrol applications was published by the author in the *Proceedings on Communications*, EUROCON '77. Since then the particular question of justifying the application of HDLC protocols, in which single bit errors may cause undetectable message errors, in real-time process control communication has been discussed intensively. Proposals for improving the error detecting properties of HDLC protocols in order to achieve data integrity and efficiency figures required in process control applications were analyzed by various experts. This paper summarizes basic results of these efforts.

I. INTRODUCTION

Many investigations into the performance of standardized data transmission protocols consider relations between information throughput, protocol parameters, and probabilities for detected message errors. The rate of nondetectable information errors in the transmission link may be considered negligible in some applications, or it is assumed that plausibility or other checks in the user layer are able to recover these errors.

However, in process control applications where short, urgent event-initiated real-time information prevails, the rate of undetectable information transmission errors and its tradeoff against transmission efficiency represent critical parameters for assessing the suitability of the chosen protocol. This analysis is restricted to assessments of undetectable information transmission errors ("residual error rates") in HDLC protocols.

A. Single Bit Errors Cause Undetectable Message Errors

The HDLC (high level data link control) protocol [1] uses the "FLAG" character: 0 1 1 1 1 1 1 0 as a frame delimiter. This bit pattern is excluded from the message frame between a leading and a trailing FLAG character by the "bit escaping mechanism." The transmitter inserts a "0" bit to the message bits, whenever five succeeding "1" bits occur in the arbitrary ("bit-oriented") text. The receiver eliminates the inserted "0" bits by discarding the "0" bits which follow five adjacent "1" bits. The end of a variable frame length is recognized in the receiving station by the trailing FLAG character.

The 16-bit frame check sequence (FCS) which terminates the frame is designed for detecting all single, double, and three bit error patterns within a frame (Hamming distance 4).

However, a single bit error may generate a spurious FLAG within a message frame. This means that a transmitted frame of B octets is received as two frames of totally $B - 1$ octets. If this error occurs and if the 16 bits ahead of the spurious FLAG are by chance a correct frame check sequence, an undetectable message error is received, as shown in Fig. 1.

The probability of this event is calculated in the next section.

II. PROBABILITY OF SPURIOUS FLAG'S IN BIT-ORIENTED FRAMES

A. Coarse Assessment of Occurrence of Spurious FLAG's

The first publication on this topic [2] estimates the probability of spurious FLAG's in the following way.

The bit escaping mechanism excludes 8 bit patterns from 256 possible octets. For the remaining 248 octets of the type

position:	1 2 3 4 5 6 7 8
octet:	0 1 1 1 0 1 1 0
bit error:	↯
FLAG:	0 1 1 1 1 1 1 0

the leading "0," all "1's," and the trailing "0" have to be transmitted error free, while the "0's" in the positions 2 to 7 have to be affected by bit errors in order to generate a FLAG octet.

In an assumed, memoryless, binary, symmetric channel model each bit position is inverted independently with the bit error probability p and is received correctly with the probability q ($q \leqslant 1 - p$, where the $<$ sign applies to binary symmetric channels subject to erasure of bits).

With these simplified model assumptions [2] calculates the probability of spurious FLAG's to be

$$P(\text{FLAG}) = (1/248)q^2\left(\binom{6}{1}pq^5 + \binom{6}{2}p^2q^4 + \cdots + p^6\right)$$

$$P(\text{FLAG}) = (1/248)q^2(1 - q^6) \qquad \text{for } q = 1 - p$$

Paper approved by the Editor for Data Communication Systems of the IEEE Communications Society for publication without oral presentation. Manuscript received February 11, 1981; revised May 27, 1981.

The author is with BBC Aktiengesellschaft Brown, Boveri & Cie., Baden, Switzerland.

0090-6778/82/0100-0252$00.75