We have illustrated our techniques for designing and implementing cyclic redundancy checks for multilevel systems by discussing a particular example where the number of levels is 9. However, the techniques that we have described in this article are applicable in general. We hope that they will be useful to engineers designing other multilevel transmission systems.

References

[1] F.J.MacWilliams, N.J.A.Sloane, *The Theory of Error-Correcting Codes*, Amsterdam: North-Holland, 1977.

[2] S.Wolfram, *Mathematica, A System for Doing Mathematics by Computer*, The Advanced Book Program, Addison-Wesley Publishing Co., Redwood City, California.

[3] R.Lidl, H.Niederreiter, *Finite Fields*, Encyclopedia of Mathematics and its Applications, vol. 20. Gian-Carlo Rota, Ed. Reading, MA: Addison-Wesley, 1983.

[4] A.R.Hammons, Jr., P.V.Kumar, A.R.Calderbank, N.J.A.Sloane, and P.Solé: The Z_4-linearity of Kerdock, Preparata, Goethals, and related codes, *IEEE Trans. Inform. Theory*, 1994, **40**, pp. 301–319

[5] J.Jedwab, S.E.Crouch, *Method and system for communicating data*, US Patent number 5,410,309, 25 Apr 1996

Efficient implementation of the shift register

We now consider how to choose a generator polynomial over Z_9 so that the shift register for computing the CRC can be implemented efficiently. We showed in the previous section that if $g(y) \in Z_3[y]$ generates a truncated code with Hamming distance 4, and $h(y)$ is any polynomial in $Z_9[y]$ whose coefficients are equivalent to those of $g(y)$ mod 3, then $h(y)$ also generates a truncated code with Hamming distance 4. When we compute the remainder of the message modulo $h(y)$ we will have to multiply by the coefficients of $h(y)$ many times, and we want this process to be as easy as possible. We can make the computation of the remainder easy by insisting that all these coefficients are either 0 or ± 1, so that the only multiplications that we do are by 0 or ± 1. This implies that given $g(y)$, we should choose $h(y)$ to look formally exactly like $g(y)$, except that the coefficients are now viewed as elements of Z_9 rather than Z_3. In our example we use the polynomial over Z_9 which is formally identical to the one found in section 2, namely $h(y) = -1 - y + y^{11} - y^{13} - y^{14} - y^{17} + y^{18}$.

We can improve the efficiency of the shift register by making multiplication modulo 9 easier. By our choice of the generator polynomial, we only have to worry about multiplication by 0 and ± 1. In order to do any computations modulo 9, we will need to convert the symbol to a binary string. This conversion is only needed at the beginning and end of the transmission. We use the following lookup chart to go back and forth between binary and 9-level symbols:

0	0000		
1	0001	-1	0010
2	0110	-2	1001
3	0100	-3	1000
4	0101	-4	1010

If we use this scheme, multiplication by minus 1 simply requires us to transpose the first two bits and the second two bits of the binary encoding. In other words, if x is encoded *abcd*, then $-x$ is encoded *badc*. This does not require any gates, (and neither does multiplication by 0 or by 1) and so we will be able to do the shift register computations in a straightforward manner. Of course, we still have to design an adder for the shift register. We need about 20–30 logic gates to do this. This is the major cost of going to a 9-level system: adding modulo 9 cannot be done with current gate technology without a sizeable number of gates.

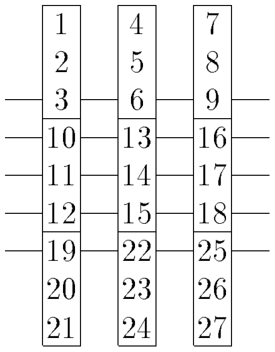

Fig. 1. Triples sent in synch, subjected to a burst error five symbols in length. Symbols 3 and 25, which are 23 positions apart, are both corrupted.

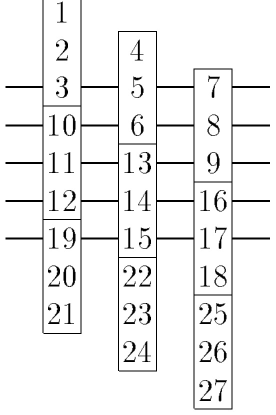

Fig. 2. Triples sent out of synch, subjected to a burst error five symbols in length. All corrupted symbols lie in a set of 17 consecutive symbols, symbols 3 to 19 inclusive.

ity that the noise that causes the burst error will affect all of the channels, and we want to devise a scheme that will enhance the burst error protection while at the same time not strongly hurting the encoding and decoding procedures. Our system was going to use three wires to transmit the data. The message would be transmitted in triples, with the first triple of symbols going on the first wire, the second triple on the second wire, and the third on the third wire. We repeat this process for the entire message. If we keep the triples in synch with eachother, then a burst error of 5 symbol periods can possibly corrupt data symbols that are 23 positions apart. We cannot guarantee that this will be detected by our CRC because it can only guarantee detection for corruption that occurs in any 18 consecutive positions. See figure 1 for a picture of this.

We can slightly modify our transmission scheme and improve the burst error protection, using an idea similar to that described in [5]. If we do not transmit the triples on the wires in synch, but we delay the second wire by one symbol period and the third wire by two symbol periods, then all the symbols corrupted by a burst of duration 5 symbol periods will be within a set of 17 consecutive positions. This will be picked up by the CRC, so we have improved the performance of the system by delaying the transmissions on the wires. See figure 2 for a picture of this.

There is essentially no cost in speed or complexity to do this, and any system which will divide the message to send it across multiple channels that could be affected by a single burst event will benefit from this staggering.

the original polynomial mod 3.

Therefore once we have used the technique of section 2 to construct a polynomial over $GF(3) = Z_3$ which generates a code truncated at length n with Hamming distance 4, any polynomial over Z_9 which reduces mod 3 to the polynomial we have constructed will also generate a code truncated at length n with Hamming distance 4.

Lemma 3: Suppose $m > 1$ and $P(x)$ is a polynomial over Z with leading coefficient 1. Suppose that $P(x) mod(m)$ generates a truncated cyclic code over Z_m of Hamming distance h. Then for all $r \geq 1$, $P(x) mod(m^r)$ generates a truncated cyclic code (truncated at the same length) over Z_{m^r} of Hamming distance at least h.

Proof: The proof is by induction on r; case $r = 1$ is trivial. Let $Q(x)$ be a polynomial over Z with degree less than the truncation length, whose coefficients lie between 0 and $m^r - 1$ and with fewer than h nonzero coefficients. Suppose that $P(x)$ divides $Q(x)$ (mod m^r). Then $P(x)$ divides $Q(x)$ (mod m), and since $P(x)$ (mod m) generates a truncated a cyclic code of Hamming distance h it follows that $Q(x)$ is equal to the zero polynomial (mod m); in other words, $Q(x) = m.Q'(x)$ for some $Q'(x)$ whose coefficients lie between 0 and $m^{r-1} - 1$. Since the leading coefficient of $P(x)$ is 1, no prime factor of m divides $P(x)$. It follows that $P(x)$ divides $Q'(x)$ (mod m^{r-1}). But $Q'(x)$ has fewer than h nonzero coefficients, and so by induction on r, $Q'(x)$ must be the zero polynomial, and hence $Q(x)$ is the zero polynomial, which completes the proof. ■

IV. Implementation issues

Burst error protection

There are two types of errors which occur in real systems. The first type are caused by random noise in the system, and the previous section discussed techniques for coding that would detect errors of this type affecting a certain number of symbols. The second type of errors are called burst errors: these occur when a sequence of consecutive symbols are compromised, typically by some external event. In general, CRCs will protect against bursts whose length is the same as the degree of the CRC because division by the polynomial $g(y)$ will yield a remainder different than the recorded remainder of the message.

In some systems, in order to increase the transmission rate, the message is not sent as a continuous stream of symbols across one channel, but it is split up into several channels. We need to consider the possibil-

```
x^9 + x^7 + x^5 + x^3 + x - 1,
x], 3]
```

yields the minimal polynomial for the primitive element that we have constructed, and is therefore a primitive irreducible polynomial of degree 17. The output of this command is $f(y) = 1 - y - y^2 - y^3 - y^4 - y^5 - y^6 - y^7 - y^8 - y^9 - y^{10} + y^{11} + y^{12} - y^{13} + y^{17}$.

When we multiply $f(y)$ by $(y-1)$, we get $g(y) = -1 - y + y^{11} - y^{13} - y^{14} - y^{17} + y^{18}$. The truncated code generated by $g(y)$ has Hamming distance 4, so it will detect any errors which result in changes to at most 3 symbols.

The same method can be used to produce other polynomials of degree 18 over GF(3) which generate a truncated code with Hamming distance 4. Given a number of such polynomials, select the one with smallest weight, because the smaller the weight the simpler the implementation of the CRC will be.

III. Lifting to a polynomial over Z_9

So far we have been discussing 3 level codes, and we would like to have a 9 level code. We could do the usual coding theory following the same model as above over $GF(9)$ rather than $GF(3)$. We could, alternatively, lift the codes from the previous sections to a code over Z_9. This alternative has the advantage of having a natural connection to the symbols that are going to be transmitted over the wires. We will follow this method of working with a 9 level code.

Recent papers in the literature (see [4]) have described how to generalize binary cyclic codes to codes over Z_4. The general idea is to follow an algorithm to get a polynomial $g(x)$ with coefficients in Z_4 so that the natural homomorphism from $Z_4[x]$ to $Z_2[x]$ maps $g(x)$ to the generator of a binary cyclic code. In [4], $g(x)$ is required to be a "basic primitive polynomial", related to its use in defining Galois Rings. This process can be mimicked to lift polynomials from Z_3 to Z_9, yielding a basic primitive polynomial with coefficients in Z_9 which when reduced modulo 3 is just the original polynomial with coefficients in Z_3. However, we do not really need to have a basic primitive polynomial, only a polynomial that has at least as good error detection as the polynomial over Z_3 from which it is lifted.

Case $m = 3, r = 2$ of the lemma below implies that the truncated code over Z_9 generated by any polynomial over Z_9 has error detection capabilities at least as good as the polynomial over Z_3 whose coefficients are obtained by reducing the coefficients of

$x^5 + x^3 + x - 1 >$ is cyclic (true of all finite fields) of order 2(1871)(34511). We need to find a generator of this cyclic group, in other words an element of this order, which we will construct by multiplying together three elements whose orders are 2, 1871, and 34511 respectively. So we need to find an element of order 34511. If $(x^2+1)^{2(1871)}$ is not 1 in the field, then it will be an element of order 34511. (If it is equal to 1, take another irreducible polynomial in place of x^2+1 and try again). To do this, first use Mathematica to compute the powers of x with the command

```
Do[g[x_,i_]:=PolynomialPowerMod[
x,i,x^17 + x^14 + x^13 + x^9 +
x^7 + x^5 + x^3 + x - 1,3],i,1,
5000]
```

then rewrite $(x^2+1)^{2(1871)}$ as
$((x^2+1)^{729})^5((x^2+1)^{81})((x^2+1)^9)$
$.((x^2+1)^7) =$
$((x^{1458}+1)^5(x^{162}+1)(x^{18}+1)(x^2+1)^7)$, exploiting the fact that raising polynomials to powers of 3 is easy mod 3; This can be computed using the Mathematica command

```
PolynomialMod[
PolynomialRemainder[
((g[x,1458]+1)^5)* (g[x,162]+1)*
(g[x,18]+1)*((x^2+1)^7), x^17 +
x^14 + x^13 + x^9 + x^7 + x^5 +
x^3 + x - 1], 3]
```

which yields the result $1+x^2+x^4+x^5+x^{11}+x^{13}-x^{14}-x^{15}$. This polynomial has multiplicative order 34511 in the finite field $GF(3)[x]/ < x^{17}+x^{14}+x^{13}+x^9+x^7+x^5+x^3+x-1 >$.

3. The element 2 has order 2 in the field. Multiplying this by x, which has order 1871, and by $1+x^2+x^4+x^5+x^{11}+x^{13}-x^{14}-x^{15}$, we get that $x^{16}+x^{15}-x^{14}-x^{12}-x^6-x^5-x^3-x$ is a primitive element of the field. If we call this element b, then the minimum polynomial for the element is
$f(y) = (y-b)(y-b^3)(y-b^9)\cdots(y-b^{3^{16}})$.
In Mathematica, call b the function

```
[x_]:=x^16 + x^15 - x^14 - x^12
- x^6 - x^5 - x^3 - x
```

and $b^{3^{n-1}}$ is the function

```
gn[x_]:=PolynomialMod[
PolynomialRemainder[
(g(n-1)[x])^3, x^17 + x^14 +
x^13 + x^9 + x^7 + x^5 + x^3 + x
- 1, x], 3]
```

4. The Mathematica command

```
PolynomialMod[
PolynomialRemainder[ (y-g1[x])
(y-g2[x]) (y-g3[x]) ...
(y-g17[x]), x^17 + x^14 + x^13 +
```

symbols. Thus, the detection scheme will be a truncated 9 level cyclic code of length at most 13500. Since the length of the packets that we will be sending is at most 13500, if we find a primitive polynomial over $GF(3)$ of degree at least 10, then by the first Lemma this will generate a truncated code with Hamming distance 3. Multiplying this primitive polynomial by $(x-1)$ will give a polynomial of degree at least 11, which by the second Lemma will generate a truncated code with Hamming distance 4 as long as we avoid codewords of the specified form. We want to organize the remainder symbols into triples like the rest of the packet, and hence we would like the degree of the primitive polynomial to be congruent to 2 mod 3. A computer search failed to find a primitive polynomial over $GF(3)$ of degree 11 or 14 which did not divide any polynomial of the form $\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j}, m < n$. (The search did not check all primitive polynomials of degree 11 and 14, so there may be one with the required property.) However we did find some primitive polynomials of degree 17 which do not divide any polynomial of this form. One example is $f(x) = 1 - x - x^2 - x^3 - x^4 - x^5 - x^6 - x^7 - x^8 - x^9 - x^{10} + x^{11} + x^{12} - x^{13} + x^{17}$.

We now describe the procedure which we followed in order to find this primitive polynomial of degree 17. Once we'd found it we used Mathematica [2] to check whether it divided any polynomial of the form $\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j}$, $m < n$; it doesn't, but if it had we would have used the same procedure to find another primitive polynomial. The procedure can be used in general to find primitive polynomials of a given degree over $GF(3)$, and is derived from the results in [3].

1. Find an irreducible polynomial of degree 17 over $GF(3)$. The multiplicative group of GF(3^{17}) is cyclic with order $3^{17} - 1$=2.1871.34511, so contains an element of order 1871. The minimal polynomial of this element (over GF_3) divides $x^{1871} - 1$, and is irreducible of degree 17. Type the command

   ```
   Factor[x^(1871) - 1, Modulus->3]
   ```

   into Mathematica. This gives 110 irreducible factors of degree 17 (and the obvious factor of (x-1)). Choose one of these; there is no clear reason at this stage to prefer any one of these to any other. The polynomial that we chose was $x^{17}+x^{14}+x^{13}+x^9+x^7+x^5+x^3+x-1$.
2. The multiplicative group of $GF(3^{17}) = GF(3)[x]/ < x^{17} + x^{14} + x^{13} + x^9 + x^7 +$

$\Rightarrow (P(x)$ divides $(1 \pm x^j)(1 \mp x^j) = x^{2j} - 1$ for some $2j < 2n < 3^d - 1)$.

But $P(x)$ is primitive of degree d, and so by standard properties of primitive polynomials does not divide $x^k - 1$ for any $k < 3^d - 1$. The result follows. ■

In order to increase the Hamming distance of the code we are using to 4 (which would detect any error altering at most 3 symbols in the packet), we can modify a standard technique from binary transmissions: when working with cyclic codes over $GF(2)$, any code whose generator is divisible by $(x+1)$ will detect any error altering an odd number of symbols in the packet. Thus, we can take a cyclic code over $GF(2)$ whose Hamming distance is 3 and construct a new code whose Hamming distance is 4 by multiplying the generator by $(x+1)$. This technique does not directly work over $GF(3)$, but the following lemma demonstrates that we can do a similar trick.

Lemma 2: Suppose $g(x)$ is a generator of a truncated cyclic code over $GF(3)$ of length n and Hamming distance at least 3, and suppose that there are no codewords of the form $\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j}$, $m < n$. Then the code of length n generated by $(x-1)g(x)$ will have Hamming distance at least 4.

Proof: Suppose that $b(x)$ is in the truncated cyclic code generated by $(x-1)g(x)$. It must be of the form $(x-1)c(x)$, where $c(x)$ is in the truncated cyclic code generated by $g(x)$. Without loss of generality, the the codeword $c(x)$ has a nonzero constant term: we can shift it if required. Since $b(x) = (x-1)c(x)$ is in the code generated by $g(x)$, we know that it cannot have fewer than 3 nonzero coefficients. Suppose (for a contradiction) that it has exactly 3 nonzero coefficients. Now $b(1) = 0$, so these coefficients must all be equal. It follows that $c(x) = b(x)/(1-x)$ is of the form $\pm(\sum_{i=1}^{k} x^{m-i} - \sum_{j=1}^{m-k} x^{m-k-j})$ for some $m < n$. But there are no codewords of this form in the truncated cyclic code generated by $g(x)$. Therefore $(x-1)c(x)$ has weight at least 4. This proves the lemma. ■

We will prove later that if $b(x)$ is the generator for a truncated cyclic code over $GF(3) = Z_3$ with Hamming distance h, then the truncated cyclic code over Z_9 whose generator is $b(x)$ (considered as a polynomial over Z_9 rather than $GF(3)$) will also have Hamming distance h.

In our application, we work with Token Ring packets, which have a maximum length of 4500 bytes (this is Token Ring). Each byte is to be encoded as a triple of 9 level

In order to ensure reliable transmission, some error detection needs to be included. This article describes how to choose a cyclic redundancy check polynomial (CRC) in such a system, and we also discuss how to implement such a system, describing a technique which increases protection against burst errors, and efficient implementation of the shift register. We will focus our attention on a system with 9 levels because of the particular application out of which this arose, but everything that we talk about can be done in a more general setting.

We assume that the reader is familiar with CRCs over $GF(2)$: see [1] for background. In section 2, we describe how to choose a CRC over $GF(3)$ with Hamming distance 4 - ie. so that error resulting in a change in up to three symbols in the packet will be detected. In section 3, we show how to use this to construct CRCs over Z_9 with Hamming distance 4. In section 4, we discuss implementation issues: we demonstrate how to send a message over three wires so that the burst error protection will be as good as possible, and we show how with an appropriate choice of CRC the shift register can be implemented in a particularly efficient way.

II. Polynomial over $GF(3)$

We will ultimately want to have error protection for a 9 level system. In order to do that, we first need to construct a system which will work for a 3 level system. We will extend this in section 4, but we state without proof at this stage that the 9 level system will have error detection capabilities at least as good as the 3 level system used to generate it. We start with an easy Lemma.

Lemma 1: Suppose that $P(x)$ is a primitive polynomial over $GF(3)$ with degree $d > 1$, and that n is an integer less than $\frac{3^d-1}{2}$. Then the code generated by P truncated at length n has a Hamming distance of at least 3.

Proof: Let $Q(x)$ be a nonzero polynomial over $GF(3)$, of degree less than n, with fewer than 3 nonzero coefficients. We want to show that $P(x)$ does not divide $Q(x)$. Clearly, $Q(x)$ must be of the form x^i or $x^i(1 \pm x^j)$ for some i, j. Since $P(x)$ is irreducible of degree greater than 1, $P(x)$ does not divide x, and hence (since the ring of polynomials over $GF(3)$ is a unique factorization domain) it does not divide x^i for any i. So we have

$(P(x)$ divides $Q(x))$

$\Rightarrow (Q(x) = x^i(1 \pm x^j)$ for some $i, j < n$ and $P(x)$ divides $(1 \pm x^j))$

Finding cyclic redundancy check polynomials for multilevel systems

James A. Davis, Miranda Mowbray and Simon Crouch

Abstract—**This article describes a technique for finding cyclic redundancy check polynomials for transmission systems which encode information in multiple voltage levels, so that the resulting redundancy check gives good protection against random errors and is efficient to implement. We discuss a way to reduce burst error in parallel transmissions, and some tricks for efficient implementation of the shift register for these polynomials. We illustrate our techniques by discussing a particular example where the number of levels is 9, but they are applicable in general.**

Keywords— **Multilevel systems, Cyclic codes, Polynomials, Communication systems**

I. Introduction

In an effort to have high speed transmission without exceeding emissions regulations, engineers have turned to transmissions which encode information in multiple voltage levels in place of the traditional binary transmission with two voltage levels.

J.A. Davis is with the Department of Mathematics and Computer Science, University of Richmond, VA 23173. This work was performed while this author was on sabbatical at Hewlett-Packard Laboratories in Bristol, England. M.Mowbray and S.Crouch are with Hewlett-Packard Laboratories, Bristol BS12 6QZ U.K.