multiplication Table

	0	1	D	D+1
0	0	0	0	0
1	0	1	D	D+1
D	0	D	D+1	1
D+1	0	D+1	1	D

B. Cyclic Codes

For algebraic codes with block length N, it will be convenient to label the N-tuples as

$$\vec{x} = [x_0, x_1, \cdots, x_{N-1}]$$

N-tuples can also be represented as polynomials of degree N-1 by the following mapping

$$\vec{x} = [x_0, \cdots, x_{N-1}] \longleftrightarrow X(D) = x_0 + x_1 D + \cdots + x_{N-1} D^{N-1}$$

Notice that for $\alpha, \beta \in GF(q)$

$$\alpha \vec{x} + \beta \vec{y} \longleftrightarrow \alpha X(D) + \beta Y(D)$$

so that the vector space of N-tuples is equivalent to the vector space of polynomials of degree $\leq N-1$.

Let the cyclic shift operator on N-tuples be $T\vec{x} = \{x_{N-1}, x_0, \cdots, x_{N-2}\}$. The corresponding polynomial is

$$\hat{T} X(D) = x_{N-1} + x_0 D + \cdots + x_{N-2} D^{N-1} = D X(D) - x_{N-1}(D^N - 1)$$

$$= [D X(D)] \bmod (D^N - 1)$$

Thus the equivalent shift operator on polynomials of degree $\leq N-1$ is multiplication by D modulo $D^N - 1$.

Def: Cyclic Code

A block code with the property that a cyclic shift of any code word is also a code word is called a cyclic code.

Properties of Linear Cyclic Codes

Let $g(D)$ be one of the monic polynomials of minimum degree such that the N-tuple $\vec{g}$ is a codeword. Assuming that $g(D)$ has degree r

$$g(D) = g_0 + g_1 D + \cdots + g_{r-1} D^{r-1} + D^r$$

$$\vec{g} = [g_0, g_1, \cdots g^{r-1}, 1, \underbrace{0 \cdots 0}_{N-r-1}]$$

any other nonzero code polynomial $a(D)$ must have deg $a(D) \geq r$. Otherwise there is a monic code polynomial $c\,a(D)$ with degree $< r$ contradicting the choice of $g(D)$. Notice that

$$\vec{g} \longleftrightarrow g(D)$$

$$T\vec{g} \longleftrightarrow D\,g(D)$$

$$T^2\vec{g} \longleftrightarrow D^2 g(D)$$

$$\vdots \qquad\qquad \vdots$$

$$T^{N-r-1}\vec{g} \longleftrightarrow D^{N-r-1} g(D)$$

are all cyclic shifts of $\vec{g}$ so that the corresponding polynomials are code polynomials. Since the code is linear $c_0 g(D) + c_1 D g(D) + \cdots c_{N-r-1} D^{N-r-1} g(D)$

$$= [c_0 + c_1 D + \cdots + c_{N-r-1} D^{N-r-1}]\, g(D) = c(D)\, g(D)$$

must be a code polynomial for any $c_0, \cdots, c_{N-r-1} \in GF(q)$
Let $\mathcal{G}$ denote the set of code polynomials. Then

$$\mathcal{G} \supset \{ c(D)\, g(D) : \deg c(D) < N-r \}$$

For any code polynomial $a(D)$,
$a(D) = Q(D)\, g(D) + r(D)$ with $\deg r < \deg g$.
Since $a(D)$ is a code polynomial $\deg a \leq N-1$,
so $\deg Q < N-r$ and $Q(D)\, g(D)$ is a codeword.
Since the code is linear $r(D)$ is also a codeword,
but the only codeword with degree $< r$
is $\vec{0}$. Therefore the set of code words $\mathcal{G}$ is

$$\mathcal{G} = \{ c(D)\, g(D) : \deg c < N-r \}$$

The set of N-tuples $\vec{g}, T\vec{g}, \cdots, T^{N-r-1}\vec{g}$ or equivalently the set of polynomials $g(D), D\,g(D), \cdots, D^{N-r-1} g(D)$ are linearly independent and span a $N-r$ dimensional vector space. Therefore the number of information symbols per codeword is

$$K = N-r \quad \text{or} \quad N-K = r = \text{no. check symbols}.$$

A generator matrix for the code is

$$G = \begin{bmatrix} g_0 & g_1 & \cdots & g_{r-1} & 1 & 0 & \cdots & 0 \\ 0 & g_0 & & & g_{r-1} & 1 & 0 \cdots & 0 \\ \vdots & & & & & & & \vdots \\ 0 & \cdots & & 0 & g_0 & \cdots & g_{r-1} & 1 \end{bmatrix} \quad \begin{matrix} \uparrow \\ \\ K \\ \\ \downarrow \end{matrix}$$

$\leftarrow N \rightarrow$

Def: $g(D)$ is called the generator polynomial of the cyclic code

Lemma: For cyclic linear codes of length N, $g(D)$ must divide $D^N - 1$.

Proof:

$$g(D) = g_0 + \cdots + D^r$$

so $D^{N-r} g(D) = D^N + \cdots + g_0 D^{N-r} = 1 \cdot (D^N - 1) + r(D)$

where $r(D) = \{ D \cdot D^{N-r-1} g(D) \} \bmod (D^N - 1)$ is the code word corresponding to the cyclic shift of $D^{N-r-1} g(D)$. Since $g(D)$ divides the left hand side, both $r(D)$ and $D^N - 1$ must contain $g(D)$ as a factor.

QED

Lemma: If $g(D)$ is any monic polynomial of degree r over $GF(q)$ that divides $D^N - 1$, the code generated by $g(D)$ is a cyclic code.

Proof:

If $X(D)$ is any codeword, then

$$D X(D) = X_{N-1} (D^N - 1) + X_{N-2} D^{N-1} + \cdots X_0 D + X_{N-1}$$

Since $X(D)$ and $D^N - 1$ are divisible by $g(D)$ $X_{N-2} D^{N-1} + \cdots + X_0 D + X_{N-1}$ is also divisible by $g(D)$ and must be a codeword

QED.

These results are summarized in the following theorem:

Theorem 7

any cyclic linear code over GF(q) with r check digits and block length N is generated by an r degree monic polynomial over GF(q) which divides D^N-1. Conversely, any r degree monic polynomial over GF(q) that divides D^N-1 generates a cyclic code of block length N with N-r information digits.

By convention N is taken to be the smallest integer such that g(D) divides D^N-1. This gives the most efficient code.

Check Polynomial

For a cyclic code generated by g(D) let

$$h(D) = (D^N-1)/g(D) = h_0 + h_1 D + \cdots + D^{N-r}$$

For any codeword $x(D) = a(D)g(D)$

$$x(D)h(D) = a(D)g(D)/h(D) = a(D)[D^N-1]$$

$$= a(D)D^N - a(D) = f(D)$$

Since $\deg a(D) \leq N-r-1$, $f_j = 0$ for $N-r \leq j \leq N-1$

Thus

$$f_j = \sum_{n=0}^{N-r} h_n x_{j-n} = 0 \quad \text{for} \quad j = N-r, \cdots, N-1$$

or

$$f_j = \sum_{l=j-N+r}^{j} x_l h_{j-l} = 0 \quad \text{for} \quad j = N-r, \ldots, N-1$$

This is a set of $r = N-K$ check equations. a check matrix for ~~this~~ code is, therefore

$$H = \overbrace{\begin{bmatrix} 1 & h_{N-r-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & h_1 & h_0 & 0 & \cdots & 0 \\ \vdots & & & & & & & \vdots \\ 0 & \cdots & 0 & 1 & h_{N-r-1} & \cdots & h_1 & h_0 \end{bmatrix}}^{r-1} \Bigg\updownarrow r$$

$$\longleftarrow \quad N \quad \longrightarrow$$

The check equations can be rewritten as

$$x_{j-N+r} = -\sum_{n=0}^{N-r-1} h_n x_{j-n} \quad \text{for } j = N-r, \ldots, N-1$$

$$N-r = N-(N-K) = K$$

or

$$x_{j-N+r} = -\sum_{\ell=j-N+r+1}^{j} x_\ell h_{j-\ell} \quad \text{for } j = N-r, \ldots, N-1$$

If $x_{N-1}, x_{N-2}, \ldots x_r$ are the information digits, these equations give a recurrence relationship for calculating $x_{r-1}, x_{r-2}, \ldots, x_0$, the check digits.

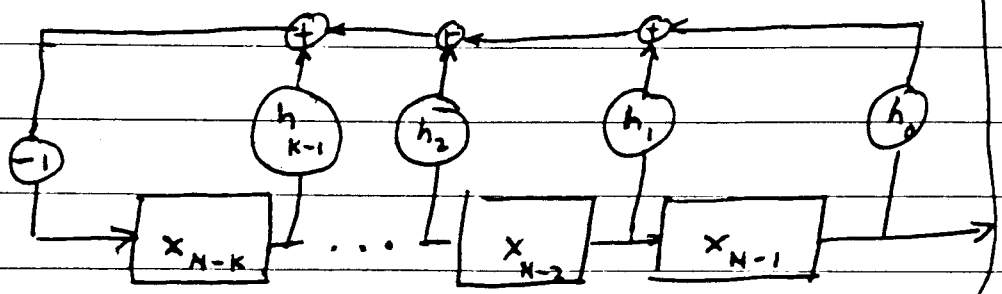
Remark: Since $h(D)$ divides $D^N - 1$, it also generates a cyclic code. This is called the dual code to that generated by $g(D)$

Encoding Circuits

Type A $K = N-r$ Stage Encoder

The device on the following page implements the recurrence relation for calculating the check

symbols. The shift register is initially loaded with the K information digits and is then shifted N times to generate the codeword.



Note: If register is shifted more than N times, the sequence repeats. If $x_0 \ldots x_{K-1} x_K \ldots x_{N-1}$ is a codeword so is $x_K \ldots x_{N-1} x_0 \ldots x_{K-1}$ because it is a cyclic shift. Each set of K info bits generates

Notice that this circuit does not correspond to G nor to the canonic systematic equivalent of G.

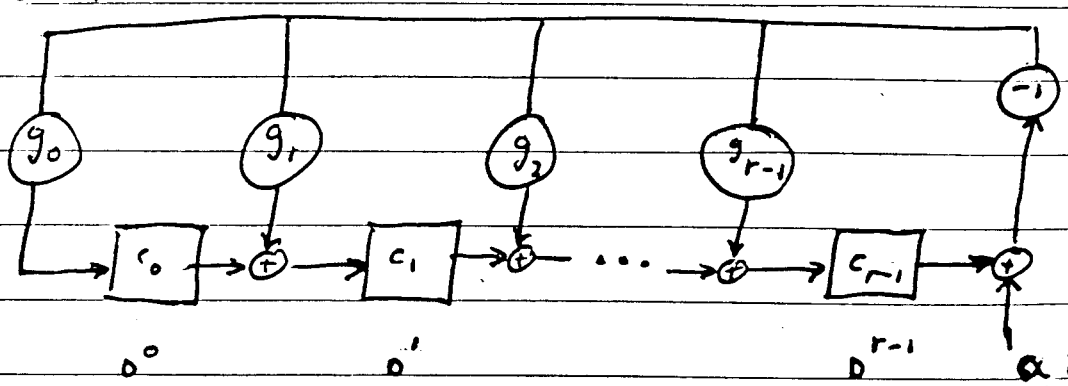
Type B $r = N - K$ stage encoder

Let the information symbols correspond to $I(D) = x_{K-1} D^{N-1} + \cdots + x_r D^r$. ($r = N-(N-r)$, $K = N-r$). By the Euclidean division algorithm $I(D) = a(D)g(D) + b(D)$ where $\deg b(D) \leq r$.

Let $X(D) = I(D) - b(D) = x_{N-1} D^{N-1} + \cdots + x_r D^r - b_{r-1} D^{r-1} - \cdots - b_0$

$= a(D)g(D)$

This is a codeword from the canonic systematic form of the code. A circuit to calculate $I(D) \bmod g(D) = b(D)$ is needed. Consider the circuit below:



The circuit has $r = N-K$ storage elements. The contents of the register can be described by the polynomial $C(D) = c_0 + c_1 D + \cdots + c_{r-1} D^{r-1}$. The input after the last stage corresponds to αD^r. Now $D^r \bmod g(D) = -g_{r-1} D^{r-1} - \cdots - g_1 D - g_0$ so

$$[D\,C(D)] \bmod g(D) = c_0 D + c_1 D^2 + \cdots + c_{r-2} D^{r-1} + [c_{r-1} D^r] \bmod g(D)$$
$$= c_0 D + c_1 D^2 + \cdots + c_{r-2} D^{r-1} - c_{r-1}[g_0 + g_1 D + \cdots + g_{r-1} D^{r-1}]$$
$$= D C(D) - c_{r-1} g(D)$$

With $\alpha = 0$, one shift of the register is seen to change its contents to just $[D\,C(D)] \bmod g(D)$. By linearity, with arbitrary α the new contents after a shift must be $[\alpha D^r + D\,C(D)] \bmod g(D)$. Suppose $\alpha = 0$ and the initial contents are $C(D)$. After one shift the contents are $[D\,C(D)] \bmod g(D)$. After another shift the contents are

$$\{D[[D\,C(D)] \bmod g(D)]\} \bmod g(D)$$
$$= \{D[D\,C(D) - c_{r-1} g(D)]\} \bmod g(D)$$
$$= [D^2 C(D)] \bmod g(D)$$

and similarly after i shifts the contents are

$$[D^i C(D)] \bmod g(D).$$

To calculate $I(D) \bmod g(D)$ the coefficients $x_{N-1}, x_{N-2}, \cdots x_{N-K}$ are shifted into the register through the final adder. If $C(D) = 0$ initially, after one shift the contents are $[x_{N-1} D^r] \bmod g(D)$, after 2 shifts $[x_{N-1} D^{r+1} + x_{N-2} D^r] \bmod g(D)$ and after $N-r$ shifts

Calculating the HDLC Frame Checking Sequence (FCS) and Error Checking at the Receiver

S.A. Tretter
June 18, 1996

1 Introduction

This report explains in detail how to calculate the HDLC frame checking sequence (FCS) and how to check for errors in the received frames. The CCITT Recommendation T.30, "Procedures for Document Facsimile Transmission in the General Switched Telephone Network," Section 5.3.7, describes the procedure as follows:

> The FCS shall be a 16 bit sequence. It shall be the 1s complement of the sum (modulo 2) of:
>
> 1. the remainer of $x^k(x^{15}+x^{14}+x^{13}+\cdots+x^2+x+1)$ divided (modulo 2) by the generator polynomial $x^{16}+x^{12}+x^5+1$, where k is the number of bits in the frame existing between, but not including, the final bit of the opening flag and the first bit of the FCS, excluding bits inserted for transparency, and
> 2. the remainder after multiplication by x^{16} and then division (modulo 2) by the generator polynomial $x^{16}+x^{12}+x^5+1$ of the content of the frame, existing between, but not including, the final bit of the opening flag and the first bit of the FCS, excluding bits inserted for transparency.
>
> As a typical implementation, at the transmitter, the initial remainder of the division is preset to all 1s and is then modified by division by the generator polynomial (as described above) on the address, control and information fields; the 1s complement of the resulting remainder is transmitted as the 16-bit FCS sequence.
>
> At the receiver, the initial remainder is preset to all 1s and the serial incoming protected bits and FCS when divided by the generator polynomial will result in a remainder of 0001110100001111 (x^{15} through x^0, respectively) in the absence of transmission errors.
>
> The FCS shall be transmitted to the line commencing with the coefficient of the highest term.

This description is confusing and vague, particularly to those not familiar with the jargon of error correcting codes. In addition, the next to last paragraph is not entirely correct. This report will attempt to clarify the procedures.

2 Details of How to Generate the FCS and Check the Received Word

Let k be the number of information bits in the frame. These are the bits between, but not including, the final bit of the opening flag and the first bit of the FCS, excluding bits inserted for flag transparency. They include address, control, and information fields. The

block length of the information bits and the FCS is $N = k + 16$. The information bits can be represented by the polynomial

$$\begin{aligned} I(x) &= x^{16}(i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \cdots + i_1 x + i_0) \\ &= i_{k-1}x^{15+k} + i_{k-2}x^{14+k} + \cdots + i_1 x^{17} + i_0 x^{16} \end{aligned} \tag{1}$$

Positions 15 down to 0 will be occupied by the 16-bit frame checking sequence.

The polynomial $x^k(x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1)$ represents a string of ones in the 16 highest order information positions.

The block diagram of a circuit for computing the remainder of a polynomial $x^{16}P(x)$ when divided by the generator polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$ is shown in Figure 1. The adders perform GF(2), also called modulo 2, addition which is the logical exclusive-or function. A box labelled D^ℓ in the figure represents an array of ℓ one-bit storage elements connected serially as a shift register. Notice that there are a total of 16 storage elements in the circuit. These will be referred to as a register and their contents represent the state of the circuit. The storage element contents or state variables are labelled $S_0(n)$ up to $S_{15}(n)$ going from left to right. The register contents at time n can be represented by the polynomial

$$S(x) = \sum_{m=0}^{15} S_m(n)x^m \tag{2}$$

The remainder when x^{16} is divided by $g(x)$ using GF(2) arithmetic and polynomial long division is found to be

$$x^{16} \bmod g(x) = x^{12} + x^5 + 1 \tag{3}$$

Notice that if the register is initially cleared to all 0's and a 1 is entered into the right-hand side through the adder where i_{k-1} is shown, the contents of the register will be $x^{12} + x^5 + 1$. Clearly, the register remains cleared if a 0 is entered. The signal $S_{15}(n)$ is treated exactly the same way. Shifting the register one position to the right with the feedback disabled is equivalent to multiplying the state polynomial by x and discarding the x^{16} term. With the feedback enabled, the $S_{15}(n)x^{16}$ term is flopped back into the register as $S_{15}(n)x^{16} \bmod g(x)$. Thus, the register state after a shift starting with the initial state, $S(n)$ and input i_n is

$$\{xS(x) + x^{16}i_n\} \mod g(x) \tag{4}$$

Combining the operations described in the previous paragraph and assuming all 0's as the initial state, it follows that the register contents after the highest order information bit, i_{k-1}, is entered are

$$i_{k-1}x^{16} \mod g(x)$$

After the next information bit, i_{k-2}, is entered the contents are

$$\{i_{k-1}x^{17} + i_{k-2}x^{16}\} \mod g(x)$$

and so on until the contents after the last bit, i_0, is entered becomes

$$\{x^{16}(i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \cdots + i_0)\} \mod g(x) \tag{5}$$

Setting the initial state of the register to a nonzero value has the effect of adding that 16-bit sequence to the highest order 16 information bits. The CCITT Recommendation specifies the initial state of all 1's corresponding to the state polynomial $x^{15}+x^{14}+x^{13}+\cdots+x^2+x+1$. Thus, the resulting state after the lowest order information bit is shifted in is

$$R(x) = \{x^k(x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1) + I(x)\} \mod g(x) \tag{6}$$

Remember that this is a polynomial of degree 15. Let $Q(x)$ be the quotient that would occur when $R(x)$ is computed. Then, according to the Euclidean division algorithm

$$C(x) = x^k(x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1) + I(x) + R(x) = Q(x)g(x) \tag{7}$$

(Remember that "+" is the same as "−" using modulo 2 arithmetic.) Notice that $C(x)$ is divisible by $g(x)$ so $C(x) \bmod g(x) = 0$.

The FCS specifed by the recommendation is the logical complement of the sequence corresponding to $R(x)$. Thus, the polynomial representation for the FCS is

$$A(x) = R(x) + x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1 \tag{8}$$

The sequence actually transmitted over the channel has the polynomial representation

$$T(x) = I(x) + A(x) = I(x) + R(x) + x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1 \tag{9}$$

The receiver contains a shift register identical to the one in the transmitter. According to the Recommendation, the initial register state is set to all 1's. Then the entire received word (excluding the bits stuffed in for flag transparency) including the 16 FCS bits is shifted into the adder on the right-hand side of the register. The Recommendation is confusing in that it neglects to specify the pre-multiplication by x^{16} caused by entering the data into the right-hand side of the register. The register state after the last FCS bit is entered is

$$\begin{aligned} P(x) &= \{x^{16}[x^k(x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1) + I(x) + R(x) \\ &\quad + x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1]\} \mod g(x) \\ &= \{x^{16}[C(x) + x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1]\} \mod g(x) \end{aligned} \tag{10}$$

Remember that $C(x)$ is divisible by $g(x)$. Therefore, if no channel errors occur, the final remainder should be

$$\begin{aligned} P(x) &= \{x^{16}[x^{15} + x^{14} + x^{13} + \cdots + x^2 + x + 1]\} \mod g(x) \\ &= x^{12} + x^{11} + x^{10} + x^8 + x^3 + x^2 + x + 1 \end{aligned} \tag{11}$$

This polynomial is equivalent to the vector [0001110100001111] specified in the Recommendation where the elements represent the coefficients of the powers of x starting with x^{15} on the left down to x^0 on the right.

The frame is checked at the receiver by comparing the actual computed remainder with the known remainder for no errors.

3 Error Detection Properties

The generator polynomial can be factored into

$$g(x) = (x+1)(x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1) \tag{12}$$

The second factor is a primitive polynomial. Thus, a cyclic code with this generator polynomial has a natural block length of $N = 2^{15} - 1$. Some of the error detection properties of this code are:

1. **Any odd number of errors is detected.**
   All codewords are a multiple of $g(x)$ and must have $x + 1$ as a factor and 1 as a root. Thus the modulo 2 sum of all the code bits must be 0. Therefore, all codewords must have an even number of 1's.

2. **All double errors are detected as long as the block length is no greater than $2^{15} - 1$.**
   A double error has the polynomial representation $x^i(x^m - 1)$, where i is an arbitrary integer and m is the distance between the two errors. The primitive factor of $g(x)$ divides $x^n - 1$ for $n = N$ but for no smaller n. Thus $g(x)$ cannot divide $x^i(x^m - 1)$ and it is not a codeword.

3. **All bursts of length 16 or less are detected.**
   A burst of length 16 or less has the form $x^i(b_{15}x^{15} + \cdots + b_1x + b_0)$. This is not divisible by $g(x)$ which has degree 16.

4. **The minimum Hamming distance between codewords is 4.**
   Codes generated by a primitive polynomial are Hamming single-error correcting codes with minimum distance 3. Thus the HDLC codewords are a subset of even weight codewords from a single-error correcting Hamming code and must have weight at least 4. The generator polynomial $g(x)$ is a codeword with weight 4.

5. **The probability of an undetected error is 2^{-16}.**
   Let K be the number of information symbols in the code. There are $N - K = 16$ check symbols. Then there are 2^K codewords. Any error pattern equal to a codeword is undetected. There are 2^N possible binary N-tuples. Thus, the undetected error probability is $2^K/2^N = 2^{-(N-K)} = 2^{-16}$.

4 Programs for Computing the FCS

Two FORTRAN programs for computing the FCS are listed at the end of this report. In both programs, the shift register state is set to all 1's and then the register is shifted 16 times. The final shift register state is the value computed at the receiver when the entire information block and FCS are shifted into the register and there are no channel errors.

The first program, HDLC.F, uses a separate array element for each state variable. This makes they algorithm structure simple and clear. However, it is not efficient for practical implementation.

The second program, HDLC1.F, saves the state variables as individual bits in a single word. The feedback shift register is implemented by shifting this word and exclusive-oring the the feedback into the word. This would be an efficient implementation with a typical microprocessor or DSP.

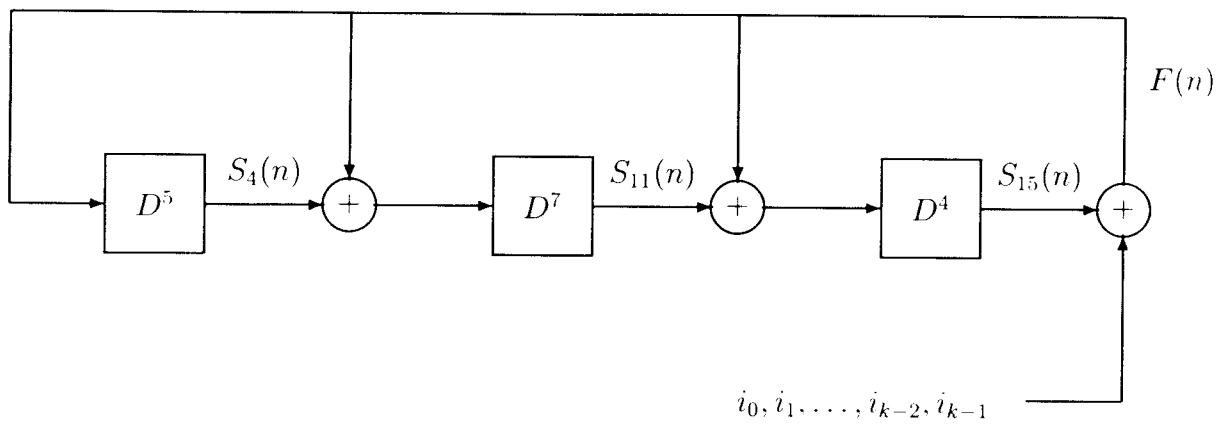


Figure 1: Feedback Shift Register to Calculate HDLC Frame Check Sequence (FCS)

5 Program Listings

Program HDLC.F

```
C     Implement HDLC encoder to compute FCS using generator
C       g(x) = x**15 + x**12 + x**5 + 1
C
        INTEGER S(0:15), F, DIN/0/
C  FCS for no channel errors
        INTEGER SDESIRE(0:15)/4*1,4*0,1,0,3*1,3*0/

C  SET INITIAL STATE TO ALL 1'S
        DO 10 I=0,15
 10       S(I) = 1
C
        DO 20 I=1,16
        F = MOD(S(15) + DIN, 2)
        S(15) = S(14)
        S(14) = S(13)
        S(13) = S(12)
        S(12) = mod(S(11) + F, 2)
        S(11) = S(10)
        S(10) = S(9)
        S(9) = S(8)
        S(8) = S(7)
        S(7) = S(6)
        S(6) = S(5)
        S(5) = mod(S(4) + F, 2)
        S(4) = S(3)
        S(3) = S(2)
        S(2) = S(1)
        S(1) = S(0)
        S(0) = F
        WRITE(*,"(1x,16I2)") (S(II),II=15,0,-1)
 20     CONTINUE
C
C  Check for no channel error
        DO 30 I=0,15
          IF(S(I).NE.SDESIRE(I)) WRITE(*,*) ' S(',I,') INCORRECT'
  30    CONTINUE
        WRITE(*,"(1x,16I2)") (S(I),I=15,0,-1)
        END
```

Program HDLC1.F

```
C   Microsoft FORTRAN 5.01
C
C      A more efficient implementation of the HDLC FCS computation
C
       INTEGER*2 STATE/16#FFFF/, MASK/16#1021/, SDESIRE/16#1D0F/
       INTEGER*2 DIN, F
C
C     STATE = [S(15),S(14), ..., S(1), S(0)]
C        STATE is preset to all 1's
C     MASK  = [0001 00001 0010 0001]
C     SDESIRE = [0001 1101 0000 1111]
C        This should be the FCS with no channel errors
C
C     Assume the input data is all 0's
       DIN = 0
C
       DO 10 I = 1,16
C        First check S(15)
          IF(IAND(STATE,16#8000).EQ.0) THEN
            F = 0
          ELSE
            F = 1
          ENDIF
C        Add in the input data
       F = MOD(F+DIN,2)
C        Update the state
        STATE = ISHFT(STATE,1)
        IF(F.EQ.1) STATE = IEOR(STATE, MASK)
C
       WRITE(*,'(1X,Z)') STATE
  10    CONTINUE
       IF(SDESIRE - STATE .EQ. 0) THEN
         WRITE(*,*) ' CORRECT FINAL STATE'
       ELSE
         WRITE(*,*) ' INCORRECT FINAL STATE'
        ENDIF
       END
```