

# The Cyclical Redundancy Check - An algorithm for error detection in data transmission and reception - By SPG Staff

---

**Introduction:** Anyone involved in sending data through physical mediums such as wire or fiber or wireless needs a way to check whether the data was received without errors. As is well known there are many ways to do this among which the simplest are parity checks. Also, error detection and coding is a vast discipline on which people have worked for years and continue to do so. It is a wonderfully mathematical process that sometimes can tax one's mind.

The cyclical redundancy check method is one of the popular methods for error detection and is used quite often. It is more involved than simple parity checks but it also provides a more powerful method of error detection. The entire process can be implemented quite simply in circuit design with simple logical elements.

This paper is a tutorial that provides a look at the CRC and how it can be implemented in logic. It is aimed at the engineer who may not have time to delve into and analyze the CRC. It simply presents how to do it quickly and simply.

**CRC Fundamentals:** The CRC procedure can be explained as follows: You have a data message you want to transmit which is  $k$  bits long. You can use the CRC to generate another sequence of bits that is  $n$  bits long. This latter sequence is called the *frame check sequence*. What you have to do is actually transmit both the original  $k$  bits of your message and the FCS that is  $n$  bits long. Therefore the total length of your transmitted message becomes  $k + n$  bits. This  $k + n$  bits should be exactly divisible by some predetermined number.

At the receiver, the received  $k + n$  bit long number is divided by the same predetermined number. If there is no remainder, then the message has been received without errors. If this division at the receiver produces a remainder then the received message has errors. It's as simple as that.

At the outset it should be understood that the CRC is based on modulo - 2 arithmetic. Modulo - 2 arithmetic uses binary addition with no carries. This is of course the same thing as the exclusive - OR operation.

For example,  $1111 + 1010 = 0101$  and  
 $11001 \times 11 = 101011$

The CRC operation can be understood using this type of arithmetic as follows:

Assume that the message to be transmitted is composed of  $k$  bits of data and  $n$  bits of FCS. Thus the message frame will have  $(k + n)$  bits, where  $n < k$ .

The k bit data is M and the n bit FCS is F. Also as mentioned above we need a predetermined number with which we will divide the complete message of k + n bits. This is P which is one more bit than the FCS or n + 1 bits long.

The CRC indicates that the quotient of T/P must have no remainder. Now, if we are to send a message composed of k bits of data and n bits of the FCS, we will have to shift the k bits of data left by n bits and include the FCS. Thus:

$$T = 2^n M + F$$

i.e. shift M by n to the left ( or multiply M by  $2^n$ ). Therefore it should be clear that the transmitted message is simply the k bit data shifted left by n bits and then concatenated with the FCS of n bits.

This frame T should be exactly divisible by the predetermined number P. Or,

$$2^n M/P = Q(\text{quotient}) + R(\text{remainder})/P \quad \text{Eqn(1)}$$

Note that the division is binary division. For binary division, the remainder is always one bit less than the divisor.

*The remainder of this division operation is the FCS.*

Therefore the message to be transmitted is:

$$T = 2^n M + R$$

In order to check if this operation satisfies the criterion of the CRC operation, consider the following:

$$T/P = (2^n M + R)/P$$

Substitute Eqn(1).

$$T/P = Q + R/P + R/P$$

Now, R/P is a binary number. Also remember in CRC operations we use Modulo - 2 arithmetic. Therefore if we add R/P to itself, Modulo - 2 arithmetic will yield a 0!

Thus:  $T/P = Q$ .

There is no remainder so the message T is exactly divisible by P. Note how easily the FCS has been generated. i.e.

Divide  $2^n M$  by P and use the remainder as the FCS! At the receiver the same division will be done on the received message. If the remainder is a 0 all is well, if not there are errors.

The pattern P, the divisor is chosen to be one bit longer than the FCS. The bit pattern for P is dependant on the type of errors to be expected. At the very least, both the high order and the low order bits of P should be 1.

In practice and in jargon, the design of CRC circuits uses polynomials. All values are expressed as polynomials in a dummy variable X with binary coefficients.

For example if M = 110011, we can say that this is a polynomial,  $M(X) = X^5 + X^4 + X + 1$ . Similarly if P = 11001, then  $P(X) = X^4 + X^3 + 1$ . All arithmetic operations are, as before Modulo - 2.

The CRC process can then also be expressed as:

$$X^n M(X) / P(X) = Q(X) + R(X) / P(X)$$

$$T(X) = X^n M(X) + R(X)$$

An error E(X) will only be undetectable if it is divisible by P(X). In practice:

*All single bit errors are detectable*

*All double bit errors are detectable*

*Any odd number of errors are detectable, as long as P(X) has a factor with at least three terms*

*Any burst error for which the length of the burst is less than the length of the FCS.*

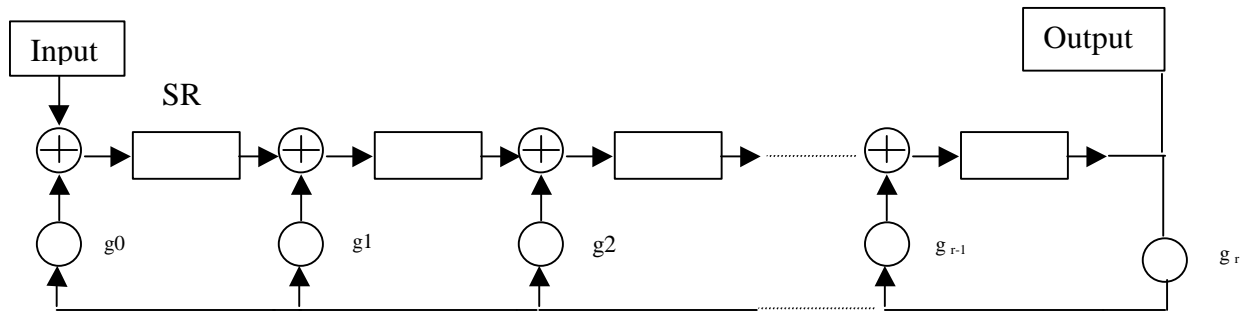
*Most larger burst errors.*

There are a few patterns of P(X) that are widely used. These are:

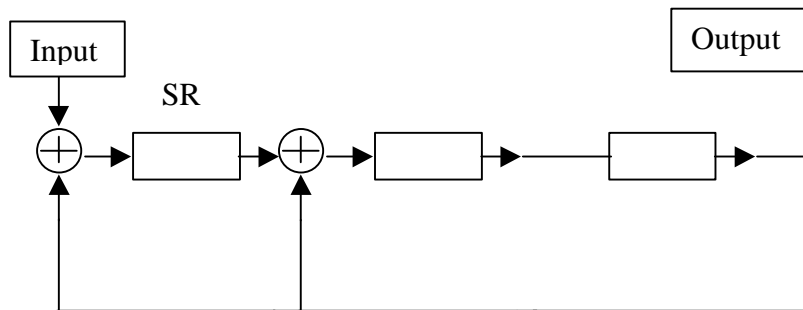
$$\begin{aligned} \text{CRC 12} &= X^{12} + X^{11} + X^3 + X^2 + X + 1 \\ \text{CRC 16} &= X^{16} + X^{15} + X^2 + 1 \\ \text{CRC - CCITT} &= X^{16} + X^{12} + X^5 + 1 \end{aligned}$$

**Polynomial Division Circuits:** As can be seen from the above, the CRC process uses polynomial division for its operating regime. Therefore, in order to implement circuitry to generate CRC one must generate polynomial division logic and circuitry. This section describes polynomial division circuits.

The simplest polynomial division circuit is shown below:



The circles with the plus signs in them are exclusive - or gates. The rectangular boxes are shift register stages.  $g_0$  through  $g_r$  are weights [0,1]. For example if the polynomial divisor is:  $1 + X + X^3$  there will be 3 shift register stages,  $g_0 = 1$ ,  $g_1=1$  and  $g_3=1$ . There will be exclusive - or gates between the input and the first SR stage, and between the first and second SR stages as shown below:



The input is the polynomial to be divided. The output is the quotient. The remainder is then contained in the shift register elements after all the input coefficients have been fed in one at a time.

Therefore the CRC encoder can be implemented rather simply using exclusive - or gates and shift register elements. The algorithm has already been explained. Take the data to be sent, shift to the left by  $n$  bits and pad with 0's. Feed it into the polynomial divider. After the correct number of shifts, the FCS becomes available which is sent as the check bits along with the data. The receiver simply takes the complete message and feeds it into its own CRC machine and does the error checks.

Obviously this whole system can be very easily simulated using any logic simulator or written as a VERILOG or VHDL file.

---

This report was presented by the techteam at Signal Processing Group Inc. Signal Processing Group Inc., offers extremely cost-effective services for the design, development and manufacture of analog and wireless ASICs and modules using state of the art semiconductor, PCB and packaging technologies. For a completely no-obligation quotation please send us your requirements. Email: [spg@signalpro.biz](mailto:spg@signalpro.biz).