CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

# Implementation of (255, 223) Reed Solomon Code Encoder /Decoder

A graduate project submitted in partial fulfillment of the requirements
For the degree of Master of Science in
Electrical Engineering

By

Ahmed Alotaibi

December 2012

The graduate project of Ahmed Alotaibi is approved:

| | |
|---|---|
| Dr. Ali Amini | Date |
| Dr. Somnath Chattopadhyay | Date |
| Dr. Nagi El Naga, Chair | Date |

California State University, Northridge

Dedication

To my parents and
my brothers

Acknowledgements

I would like to thank my project chair and committee: Dr. Nagi El Naga, Dr. Ali Amini, and Dr. Somnath Chattopadhyay for their time and mentorship.

I would also like to thank my family for their love and support throughout the last two years, because I could not have finished this project without their encouragement.

Table of Contents

List of Figures

Abstract

Implementation of (255,223) Reed Solomon code

Encoder /Decoder

By

Ahmed Alotaibi

Master of Science in Electrical Engineering

In this project, the algorithm of the Reed-Solomon code encoder and decoder is discussed, followed by the implementations of (255, 223) Reed-Solomon codes. The encoder takes a 223-byte date block and generates a 255-byte code block to be transmitted on a digital communication channel. This code is defined over a Galois Field GF ($2^8$) and has the capability of correcting up to sixteen short bursts of errors. The software part of the design is written in C language and tested using Simulink, and the hardware part is implemented using a microcontroller.

The encoding process is simpler than the decoding process and requires less complex hardware. The decoder, however, is implemented using a parallel structure to speed up the Galois Field computations necessary to find out the locations of the errors and their values.

The decoder logic circuit comprises multiple units in order to perform a parallel computation for the syndrome calculation. The Berlekamp's iterative algorithm is used to determine the coefficients of the error location polynomial. At the same time, the Chien's searching algorithm is used to calculate the roots of the error location polynomial, their error values, and perform the correction of the found errors.

# Chapter 1

# Introduction

## 1.1. Introduction

In 1948, Claude Shannon, considered to be the founding father of electronic communications, began to prove that codes with special properties can accomplish unfailing data communications over a noisy channel, if the capacity of the channel is larger than the data rate of the message source. This proof has opened the door to further discoveries in that field, where substantial work has been done not only in digital communications, but also in error control coding theory. Since that time, many engineers have been trying to construct such code to prove the Shannon theorem without any mathematical abstracts. The engineers have ended up, however, being able to detect the error but not correct it; one code appending the parity check bit to the message bits results in a single bit error.

The demands of achieving reliable coding have continued over time due to the high volume of data exchanged. The new code needed to be constructed on form where it has enough redundancy to be able to detect, or detect and correct, multiple bit errors up to a definite probability that relates to the characteristics of the digital channel. The code rates (message length/ code length in bits) are close to unity and are easy to implement in hardware.

The cyclic code is one of the codes that has been developed and has become a useful code in data communications. This code is very reliable and relatively easy to implement. In practical applications, the cyclic code is used in systematic forms which have the advantage of the parity data, which can simply be appended to the source block, and receivers do not need to do any work to recover the original source symbols if received correctly.

The Bose-Chaudhuri-Hocquenghem (BCH) codes are a large class of cyclic code. The BCH comes in two forms: binary BCH code and nonbinary BCH code. Among the nonbinary

BCH code, the Reed Solomon code contains code symbols that are elements of a finite field (Galois Field) with an order of $2^m$. All information, voice, video, pictures, and data are represented by a long string of zeroes and ones, also known as bits. All digital systems use binary symbols at the hardware level to represent any type of data; therefore, the representation of elements from GF ($2^m$) is straightforward.

**1.2. Objective**

The objective of this project is to design and implement an error detection and correction system using Reed Solomon code. The first part discusses the general procedure for encoding and decoding Reed Solomon codes.

The block code length in the code has been selected as 255 and the message length is 223, which is known as (255, 223) Reed Solomon codes. The code rate is 223/255 = 0.875 and this code is defined over GF ($2^8$), where each symbol is represented by an 8-bit (one byte) to make 255 x 8 = 2040 binary bits transmitted serially. This code is capable of detecting and correcting up to sixteen errors bytes in every block of 255 bytes. In other words, errors in up to 16 bytes anywhere in the codeword can be automatically corrected. The choice over GF ($2^8$) is justified by the large spread of the popular 8-bit data buses used in the industry today. This code is not hard to implement in hardware which is not a part of this project. However, testing the code in software using both Simulink and C language makes this code more reliable and the most popular code used for any type of data communication.

### 1.3. Outline

The project consists of eight chapters and three appendixes. The second chapter is an introduction to information theory and coding. There is also a brief introduction about Hamming code as well as Reed Solomon code. Please note that the basic algebraic field structure should be understood by the reader. Chapter three contains the theory and the theoretical description of the Reed Solomon code encoder and decoder. Chapter four presents the general concept of the project from the hardware point of view. Chapter five is the implementation of the encoder and chapter six is the implementation of the decoder. Chapter seven includes the test of the design and an explanation of the use of simulation of (255, 223) Reed Solomon code. Chapter eight is the conclusion. The appendixes contain assisting materials.

The simulation software used in this project is Simulink, representing the project as blocks where the parameters are specified. The Matlab is used to generate nonzero elements in the field and C language source code is used in the implementation of the encoder and the decoder.

## Chapter 2

## Hamming codes

### 2.1. Hamming codes

The early 1950s is the beginning of error correcting codes. Richard W. Hamming discovered the first class of linear block code for error correction, which was later named after him and is known as Hamming code. Hamming code is a set of error-correction codes used to detect and correct bit errors occurring in digital communication or data storage systems and can be defined for any positive integer m ≥3 having the following properties:

Code length $n = 2^m-1$

Number of parity check bits $n-k = m$

Number of information symbols $k = (2^m–1) – m$

Minimum distance $d_{min}= 3$

Error correcting capability $t =1$

The Hamming code's single-error-correcting (SEC) and double-error-detecting (DED) extended versions marked the beginning of coding theory. These codes remain important for both theoretical and practical reasons.

### 2.2. BCH codes

The generalization of the Hamming codes for multiple error correction codes is known as BCH codes, named and discovered independently by A. Hocquenghem in 1959 and R.C Bose and D.K. Ray-Chaudhuri in 1960. BCH codes are cyclic codes and are, by far, the most extensive and powerful codes. BCH codes are first defined in binary symbols and then generalized to codes in $p^m$ for any codes where p is a prime number and m is any positive

number. In this section, binary BCH codes will be described and the following section will address nonbinary BCH code.

For any positive integer m and t where $t < 2^m-1$, there are binary BCH codes which have these properties:

Block length $n = 2^m-1$

Number of parity check digits $n - k \leq mt$

Minimum distance $d_{min} \geq 2t + 1$

Binary BCH codes are capable of correcting any combination of t or fewer errors in a code block of $n = 2^m-1$ digits, which is a t-error correcting BCH codes.

Letting α be a primitive element in GF ( $2^m$) and consider the following sequence:

$$\alpha^1, \alpha^2, \alpha^3, \ldots\ldots\ldots, \alpha^{2t}$$

If $m_i(x)$ is the minimum polynomial of $\alpha^i$ , then the generator g(x) polynomial of the t-error correcting BCH code is the least common multiple of $m_1(x), m_2(x)\ldots\ldots, m_{2t}(x)$

$$\text{i.e. } g(x) = \text{LCM } \{m_1(x), m_2(x), \ldots\ldots\ldots, m_{2t}(x)\}$$

$\alpha^1, \alpha^2, \alpha^3, \ldots\ldots\ldots, \alpha^{2t}$ respectively, they are roots of $g(\alpha^i) = 0$. If i is an even integer, then i'= i' $2^j$ where i' is an odd integer and $j \geq 1$, and they have the same polynomial i.e. $m_i(x) = m_{i'}(x)$ which means that g(x) for t-correcting errors can be reduced to

$$g(x) = \text{LCM } \{m_1(x), m_2(x), \ldots\ldots\ldots\ldots, m_{2t}(x)\}$$

where the degree of any minimum polynomial is m or less and the degree of g(x) is mt or less; if t is small, then n-k exactly equals mt.

### 2.3. Nonbinary BCH codes and Reed Solomon codes

Nonbinary BCH codes can be defined as, if p is a prime number and q is any power of p, then codes with code symbols from the elements of Galois Field GF (q) exist. These codes are called q-ary codes and for any positive integers s and t there exist a q-ary BCH code with block length $n = q^s-1$ which can correct any combination of t or fewer errors and require no more than 2st parity check bits. However, when s = 1 is special subclass of q-ary BCH and known as Reed Solomon codes. Reed Solomon codes are t-error correction codes and have the following properties:

| | |
|---|---|
| Block length | n = q - 1 |
| Number of parity bits | n - k = 2t |
| Dimension | k = q -1-2t |
| Minimum distance | d = 2t+1 |

Letting $q = 2^m$ then the generator polynomial of a primitive t error correcting Reed Solomon codes will be of length $2^m-1$ and can be expressed as:

$$g(x) = (x+ \alpha)(x+\alpha^2)\ldots\ldots(x+\alpha^{2t})$$

Where α is a primitive element of GF ( $2^m$).

In Reed Solomon codes, the length of the code is one less than the size of the code alphabet and the minimum distance is one greater than the number of parity check symbol n-k. So, correcting t symbol errors requires no more than 2t parity symbols. Reed Solomon codes, constructed by Reed and Solomon in 1960, are therefore very effective in correcting random symbol errors and random burst errors in communication, as well as data storage systems, ranging from deep space telecommunications to compact discs.

## Chapter 3

## Reed Solomon encoder/decoder

### 3.1. Reed-Solomon encoding

The generating polynomial for an R-S code is driven as:

$$g(X) = g_0 + g_1X + g_2X^2 + \ldots + g_{2t-1}X^{2t-1} + X^{2t}$$

With degree equal to the number of parity symbols (n-k). Since the generator polynomial of degree n-k, there must be precisely n-k successive powers of α ($\alpha, \alpha^2, \alpha^3, \ldots\ldots\ldots, \alpha^{2t}$) that are roots of the polynomial. It is not necessary to start with the root α; starting with any power of α is possible. Reed Solomon codes are cyclic codes and encoding in systematic form is analogous to the binary encoding procedure. By shifting a message polynomial m(X), which is to be encoded into the rightmost k stages of a codeword register, and then appending a parity polynomial, p(X) by placing it in the leftmost n-k stages. Then, multiplying m(X) by $X^{n-k}$ is done to manipulate the message polynomial algebraically so that it is right-shifted n - k positions. Next, dividing $X^{n-k}$ m(X) by the generator polynomial g(X), which can be written as:

$$X^{n-k}\, m(X) = q(X)\, g(X) + p(X)$$

Where q(X) is the quotient polynomial and p(X) is the remainder polynomial with degree of 2t-1 or less. This is the same as in the binary case where the remainder is the parity which can be written as:

$$P(X) = X^{n-k}\, m(X) \bmod g(X)$$

So, the resulting codeword polynomial U(X) can be written as:

$$U(X) = P(X) + X^{n-k}\, m(X)$$

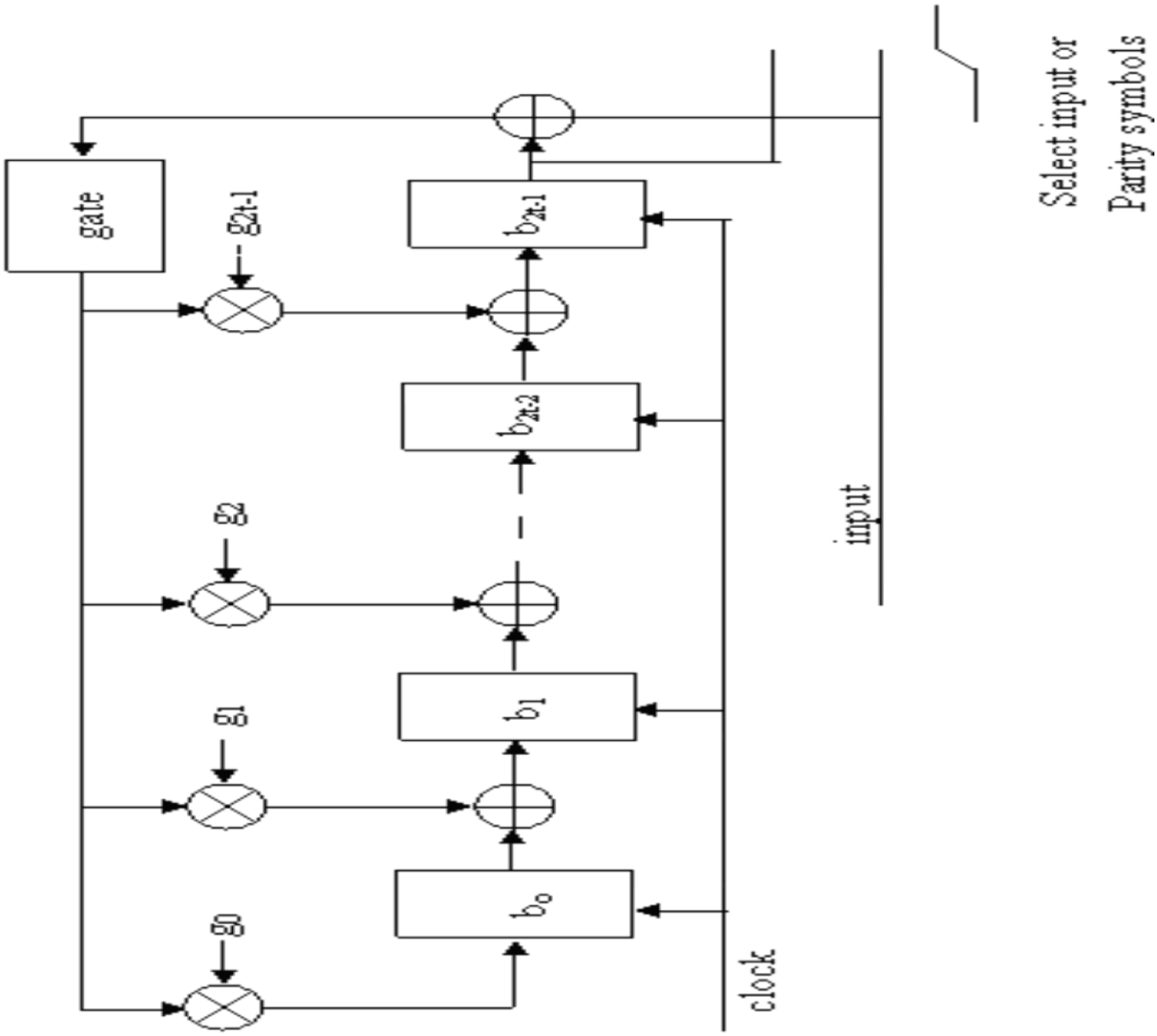



Figure 3 .1 Reed Solomon code encoder block diagram

### 3.2. Reed-Solomon decoding

Let

$v(X) = v_0 + v_1X + \ldots\ldots + v_{n-1}X^{n-1}$ is the transmitted code vector and let

$r(X) = r_0 + r_1X + \ldots\ldots + r_{n-1}X^{n-1}$ is the received vector

This assumes that, during transmission, the codeword becomes corrupted so that some symbols are received in error (this number of errors corresponds to the maximum error-correcting capability of the code). Then, the error pattern e(X) caused by the channel noise during the transmitter can be written as:

$$e(X) = r(X) - v(X)$$

$$e(X) = e_0 + e_1X + \ldots\ldots + e_{n-1}X^{n-1} = \sum_{i=0}^{n-1} e_iX^i$$

In binary BCH code decoding, the decoder only needs to find the error locations. However, the nonbinary BCH codes and R-S codes symbols require not only the identification of the error locations, but also determining the correct symbol values at those locations. So, the decoder will calculate the syndrome from the received vector r(X), which is the first step of decoding R-S process. The syndrome S with 2t components can be defined as the following:

$$S_i = r(\alpha^i)$$

$$S_i = r_0 + r_1\alpha^i + \ldots\ldots + r_{n-1}(\alpha^i)^{(n-1)} \qquad \text{for } i = 1, 2, \ldots, 2t$$

$S_i = v(\alpha^i) + e(\alpha^i)$ from the above equations for i = 1, 2,….,2t

The roots of the code polynomial v(X) are $\alpha, \alpha^2, \ldots\ldots\ldots, \alpha^{2t}$

Therefore, $v(\alpha^i) = 0$ for i = 1, 2,…., 2t which makes the syndrome equation written as:

$$S_i = e(\alpha^i) \qquad \text{for } i = 1, 2, \ldots, 2t$$

Assuming e(X) is an error pattern of z errors then,

$$e(X) = X^{j1} + X^{j2} + \ldots\ldots + X^{jz} \qquad (2.1)$$

And 2t syndrome equations have been formed as following:

$$S_1 = \alpha^{j1}+\alpha^{j2}+\ldots\ldots+ \alpha^{jz}$$

$$S_2 = (\alpha^{j1})^2 + (\alpha^{j2})^2 + \ldots\ldots..+ (\alpha^{jz})^2$$

$$\vdots$$

$$S_{2t} = (\alpha^{j1})^{2t} + (\alpha^{j2})^{2t}+\ldots\ldots...+ (\alpha^{jz})^{2t}$$

Any error correction process is a method of solving the above equations and finding $\alpha^{j1}$, $\alpha^{j2}$,……., $\alpha^{jz}$then the powers $j_1, j_2,\ldots., j_z$ which are the error locations in e(X) at eq.(2.1). These above equations have many finite possible solutions and each one will yield a different error pattern. When the number of errors in the actual error pattern e(X) is t or less ($z \leq t$), then the solution with the smallest number of errors is the correct solution and the error pattern corresponding to that solution will be the actual error pattern. On the other hand, for large t, solving the above equations will be more difficult and ineffective. However, determining $\alpha^{jl}$ from the syndrome components $S_i$ is done as the following:

Let

$$\beta_l = \alpha^{jl} \qquad \text{for } 1 \leq l \leq z$$

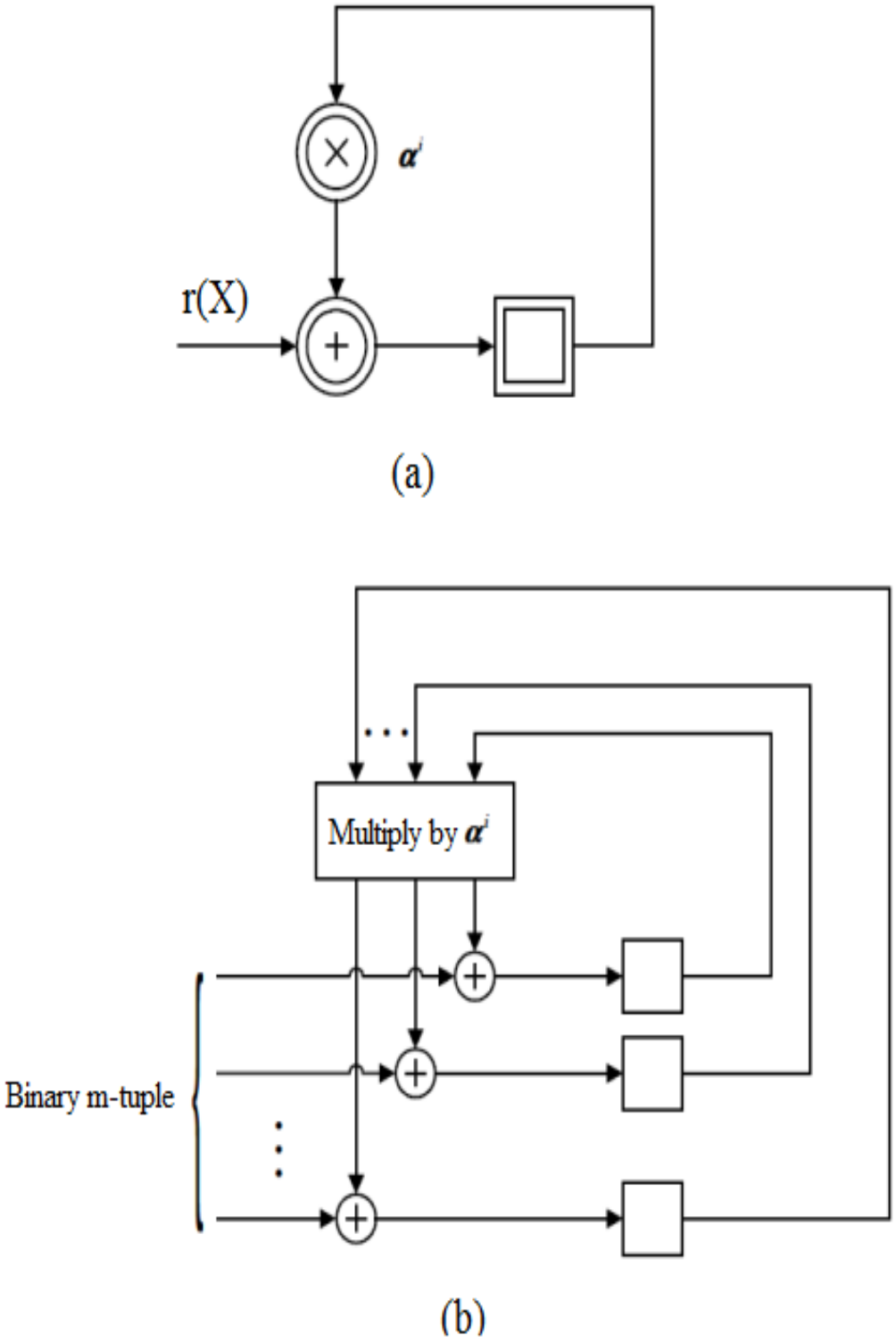
Be the error location numbers which means the 2t syndrome equations above are symmetric functions in $\beta_1, \beta_2,\ldots\ldots,\beta_z$ . This is also known as power sum symmetric functions and can be written as:

$$S_1= \beta_1+\beta_2+\ldots\ldots\ldots+\beta_z$$

$$S_2= \beta_1^{\,2} +\beta_2^{\,2} +\ldots...+\beta_z^{\,2}$$

$$\vdots$$

$$S_{2t}= \beta_1^{\,2t}+\beta_2^{\,2t}+\ldots\ldots+\beta_z^{\,2t}$$




Figuer 3.2.Syndrome computation for Reed Solomon code in
(a)- GF($2^m$) and (b)- binary representation form

Next, the error location polynomial $\sigma(X)$ is defined as the following:

$$\sigma(X) = (1 +\beta_1 X)(1 +\beta_2 X)\ldots\ldots\ldots\ldots\ldots\ldots\ldots..(1+\beta_z X)$$

$$\sigma(X) = \sigma_0+\sigma_1 X+\ldots\ldots\ldots\ldots\ldots\ldots+ \sigma_z X^z$$

Where

$$\sigma_1= 1$$

$$\sigma_2 = \beta_1+ \beta_2+\ldots\ldots\ldots+\beta_z$$

$$\vdots$$

$$\sigma_z= \beta_1\beta_2\ldots\ldots\ldots\beta_z$$

The roots of $\sigma(X)$ are the inverses of error location numbers: $\beta_1^{-1}, \beta_2^{-1}, \ldots\ldots\ldots, \beta_z^{-1}$, so the $\sigma_i$'s are elementary symmetric functions of $\beta_z$'s can be related to the syndrome components $S_i$'s by the following Newton identities:

$$S_1+\sigma_1 =0$$

$$S_2+ \sigma_1 S_1+ 2\sigma_2 =0$$

$$\vdots$$

$$S_{z+1}+ \sigma_1 S_z+\ldots\ldots\ldots\ldots..+\sigma_{z-1} S_2+ \sigma_z S_1=0$$

At this point, the error correcting procedure for R-S codes decoder has these four steps:

1. Compute the syndrome $S = S_1 , S_2 , \ldots\ldots , S_{2t}$ from the received polynomial r(X).
2. Determine the error location polynomial $\sigma(X)$ from the syndrome components.
3. Determine the error location numbers$\beta_1, \beta_2,\ldots\ldots\ldots, \beta_{2t}$ by finding the roots of $\sigma(X)$.
4. Substitute the error location numbers into the error polynomials and solve for the corresponding error values $e_{zi}$ . Knowledge of the values of $\beta_i$ and $e_{zi}$ is sufficient for error correction.

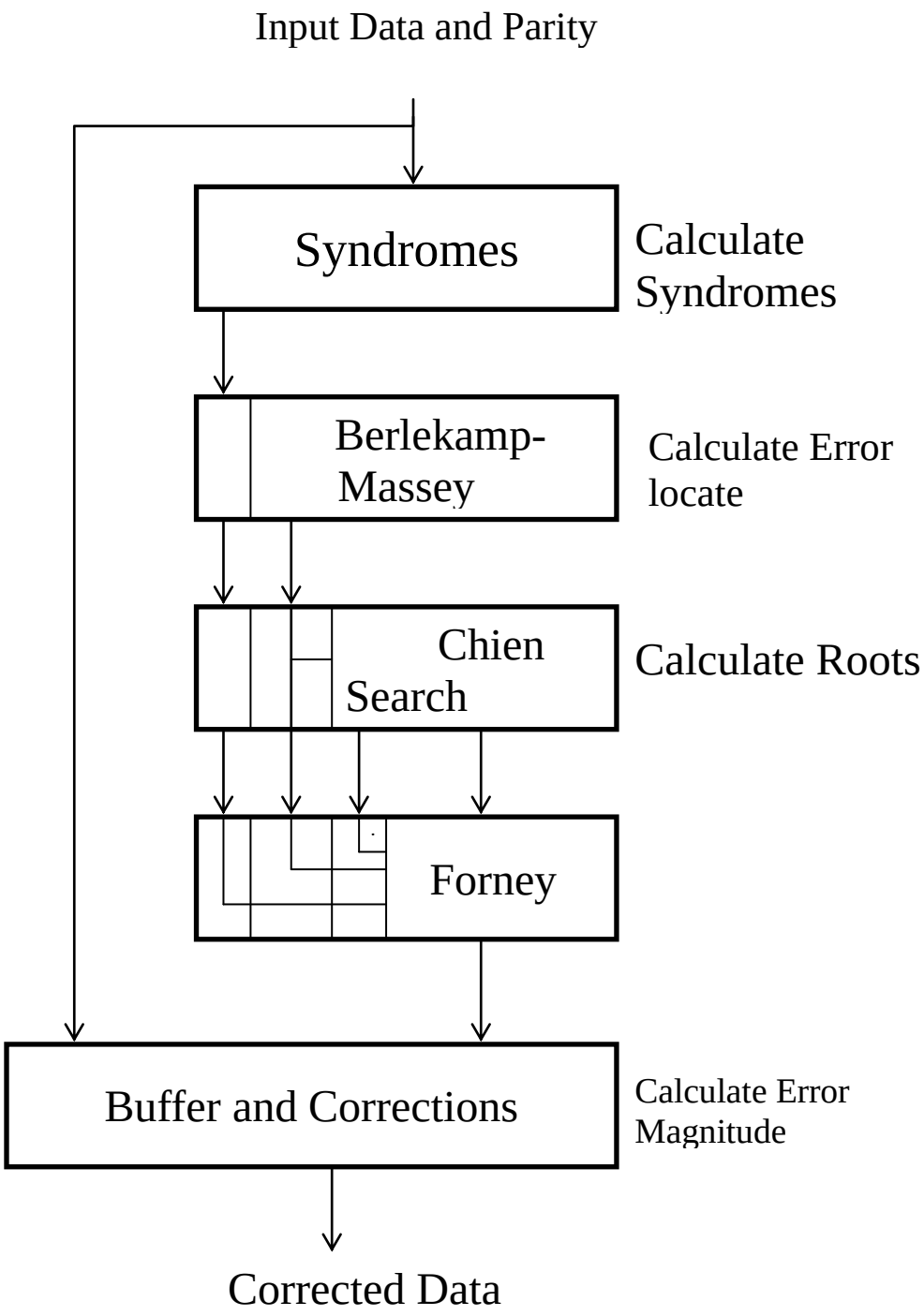



Figure 3.3 Reed Solomon decoder block diagram

## Chapter 4

## System specification

### 4.1. System specification

The system will have (255, 223) Reed Solomon code where the symbols of the code are elements of GF ($2^8$) and have the following characteristics:

Degree of the polynomial m = 8

Block length n = $2^8$-1 = 255

Number of parity check digits n - k = 2t = 32

Error correction capability t = 16

However, each symbol is represented by eight binary digits or one byte. Also, each data block contains 223 information symbols. This code is capable of correcting up to sixteen short burst errors of one byte or any burst error combination of up to a total length of eight bytes, providing that they only affect a maximum of sixteen individual symbols.

The elements of GF ($2^8$) are generated by primitive polynomial of degree 8:

$$p(X) = 1+X^2+X^3+X^4+X^8$$

let $\alpha$ be the primitive element in GF($2^8$) and the root of p(X)

then $$p(X) = 1+X^2+X^3+X^4+X^8 = 0$$

or

$$p(X) = 1+X^2+X^3+X^4 = X^8$$

So, the elements can be represented in an 8-tuple with 8 components being 0 or 1 and represent code word. The zero element of GF($2^8$) appears as an all zero 8-tuple. Matlab is used to represent nonzero elements as shown in Appendix A.

Also, if α is a primitive element in GF($2^m$), then the root of p(x) is only the first thirty-two powers of α and are the roots of the generator polynomial. Meanwhile, the generator polynomial for (255, 223) code is given by:

$$g(X) = (X+\alpha)(X+\alpha^2)(X+\alpha^3)(X+\alpha^4)(X+\alpha^5)(X+\alpha^6)(X+\alpha^7)(X+\alpha^8)(X+\alpha^9)(X+\alpha^{10})$$
$$(X+\alpha^{11})(X+\alpha^{12})(X+\alpha^{13})(X+\alpha^{14})(X+\alpha^{15})(X+\alpha^{16})(X+\alpha^{17})(X+\alpha^{18})(X+\alpha^{19})$$
$$(X+\alpha^{20})(X+\alpha^{21})(X+\alpha^{22})(X+\alpha^{23})(X+\alpha^{24})(X+\alpha^{25})(X+\alpha^{26})(X+\alpha^{27})(X+\alpha^{28})$$
$$(X+\alpha^{29})(X+\alpha^{30})(X+\alpha^{31})(X+\alpha^{32})$$

$$g(X) = 45+216X+239X^2+24X^3+253X^4+104X^5+27X^6+40X^7+107X^8+50X^9+163X^{10}+210X^{11}$$
$$+227X^{12}+134X^{13}+224X^{14}+158X^{15}+119X^{16}+13X^{17}+158X^{18}+X^{19}+238X^{20}+164X^{21}$$
$$+82X^{22}+43X^{23}+15X^{24}+232X^{25}+246X^{26}+142X^{27}+50X^{28}+189X^{29}+29X^{30}+232X^{31}+X^{32}$$

therefore, the coefficients of g(x) used in the encoder multiplication are :

$g_0 = \alpha^{18} = 45$, $g_1 = \alpha^{251} = 216$, $g_2 = \alpha^{215} = 239$, $g_3 = \alpha^{28} = 24$, $g_4 = \alpha^{80} = 253$,

$g_5 = \alpha^{107} = 104$, $g_6 = \alpha^{248} = 27$, $g_7 = \alpha^{53} = 40$, $g_8 = \alpha^{84} = 107$, $g_9 = \alpha^{194} = 50$,

$g_{10} = \alpha^{91} = 163$, $g_{11} = \alpha^{59} = 210$, $g_{12} = \alpha^{176} = 227$, $g_{13} = \alpha^{99} = 134$, $g_{14} = \alpha^{203} = 224$,

$g_{15} = \alpha^{137} = 158$, $g_{16} = \alpha^{43} = 119$, $g_{17} = \alpha^{104} = 13$, $g_{18} = \alpha^{137} = 158$, $g_{19} = \alpha^{0} = 1$,

$g_{20} = \alpha^{44} = 238$, $g_{21} = \alpha^{149} = 164$, $g_{22} = \alpha^{148} = 82$, $g_{23} = \alpha^{218} = 43$, $g_{24} = \alpha^{75} = 15$,

$g_{25} = \alpha^{11} = 232$, $g_{26} = \alpha^{173} = 246$, $g_{27} = \alpha^{254} = 142$, $g_{28} = \alpha^{194} = 50$, $g_{29} = \alpha^{109} = 189$,

$g_{30} = \alpha^{8} = 29$, $g_{31} = \alpha^{11} = 232$, $g_{32} = 1$

At the same time, a Reed Solomon code with smaller dimensions can be chosen. (255,223) code has a suitable symbol size that matches the 8-bit data byte commonly used today. Additionally, the length of this code of 255 symbols can be shortend to any length so that it can match any system specification without any major change in the encoder/decoder hardware circuitry.

### 4.2.Encoder architecture

Encoders are designed as feedback shift register, Figure 3.1. The data massage blocks of 223 symbols shift sequentially as an input to the encoder and when the last message symbol is loaded, the feedback register contains the thirty-two parity check symbols. These symbols will then be shifted out following the 223 information symbols to generate a code word of 255 symbols as an output of the encoder.

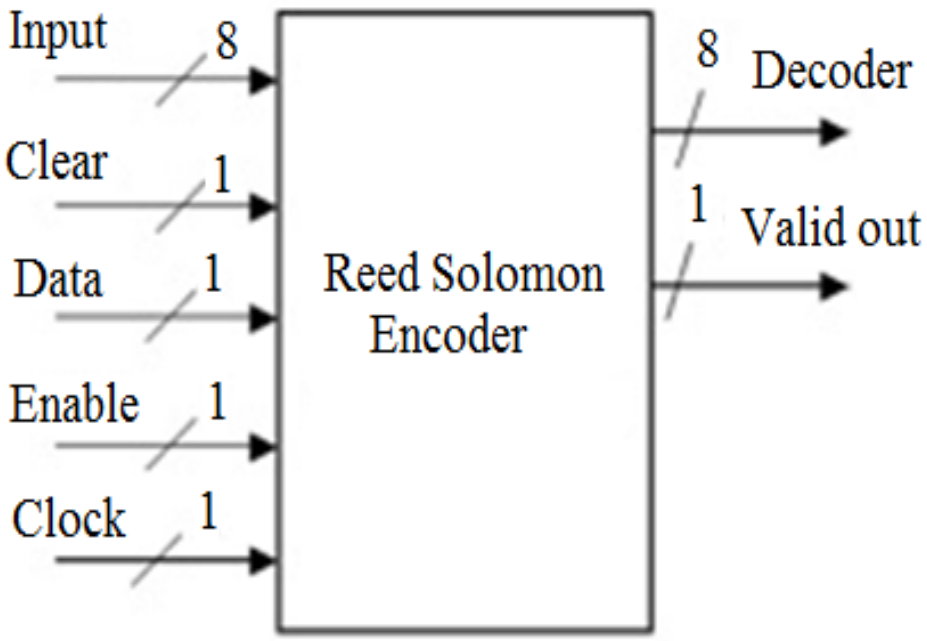



Figure 4.1 Reed Solomon encoder block diagram

### 4.3.Decoder architecture

The decoder operation is carried out as follows:

1- Compute the syndrome S = ($S_1$, $S_2$ , …… , $S_{2t}$).

In this step, (255,223) Reed Solomon code syndrome calculation can be done as follows:

Let r(α) be the received code polynomial. Then,

$$r(\alpha) = S_1 = e_{j1}\alpha^{j1} + e_{j2}\alpha^{j2} + \ldots\ldots + e_{jz}\alpha^{jz}$$

$$r(\alpha^2) = S_2 = e_{j1}(\alpha^{j1})^2 + e_{j2}(\alpha^{j2})^2 + \ldots\ldots + e_{jz}(\alpha^{jz})^2$$

$$\vdots$$

$$r(\alpha^{32}) = S_{32} = e_{j1}(\alpha^{j1})^{32} + e_{j2}(\alpha^{j2})^{32} + \ldots\ldots + e_{jz}(\alpha^{jz})^{32}$$

All the received data block will shift into the syndrome processor using the same circuit to compute the thirty-two syndrome components $S_1$, $S_2$ , …… , $S_{32}$ which will be done in sequence form. The only difference is the multiplication values in the feedback loop for each syndrome.

2- Determine the error location polynomial σ(X).

In this step, Berlekamp's iterative algorithm method (Table 3.1.) has been used, without being given any proof, to determine the coefficients of the error location polynomial σ(X). There are (n-k) = 2t = (255-223) = 32 iterations at step μ to be performed and the algorithm will yield the minimum degree polynomial that will satisfy the first μ Newton equations.

| $\mu$ | $\sigma^{(\mu)}(X)$ | $d_\mu$ | $1_\mu$ | $\mu - 1_\mu$ |
|---|---|---|---|---|
| -1 | 1 | 1 | 0 | -1 |
| 0 | 1 | $S_1$ | 0 | 0 |
| 1 | | | | |
| 2 | | | | |
| . | | | | |
| . | | | | |
| 2t | | | | |

At any iteration step, the entries in the table start with evaluating the discrepancy value $d_\mu$ from the previous row and:

If $d_\mu = 0$, we set $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X)$, which is the entered as in the previous step.

If $d_\mu \neq 0$, a correction factor must be added. So,

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d^{-1}{}_\rho \, X^{(\mu-\rho)} \, \sigma^{(\rho)}(X)$$

Where $\rho$ is a previous iteration step such that $d_\rho \neq 0$, and $\rho - 1_\rho$ in the last column of the table have the largest value. The next entry in the last two columns are:

$$1_{\mu+1} = \max [1_\mu , 1_\rho + \mu - \rho]$$

The last entry for the current iteration step involves computing the new discrepancy value as follows:

$$d_{\mu+1} = S_{\mu+2} + \sigma_1{}^{\mu+1} S_{\mu+1} + \ldots\ldots + \sigma_{1_{\mu+1}}{}^{(\mu+1)} S_{\mu+2-1_{\mu+1}}$$

After the last step in the iteration process, which is the last row in this case or step number thirty two, the $\sigma^{(\mu)}(X)$ column has the correct error location polynomial if there are less than sixteen errors in the data block.

3- Determine the error value evaluator.

In this step, the roots of error location can be found by Chien's search, which initially starts by loading the register with the coefficients of the error location polynomial σ(X). Then, at each clock pulse, every register is multiplied by the feedback loop value. After each clock, all the registers are XOR-ed together with unity element of GF ($2^8$); if the XOR value is zero, then the root has been found. The search is run for 256 clock cycles and t+1 = 17 stages in the search which each consists of multiplexer, Adder (XOR), and a register. Meanwhile, the feedback values have been chosen to be powers of the primitive element α, such that in 254 trials, all nonzero elements of GF ($2^8$) are checked as the roots of the error location polynomial. The resulting finding of a root is saved as the inverses of the error location numbers.

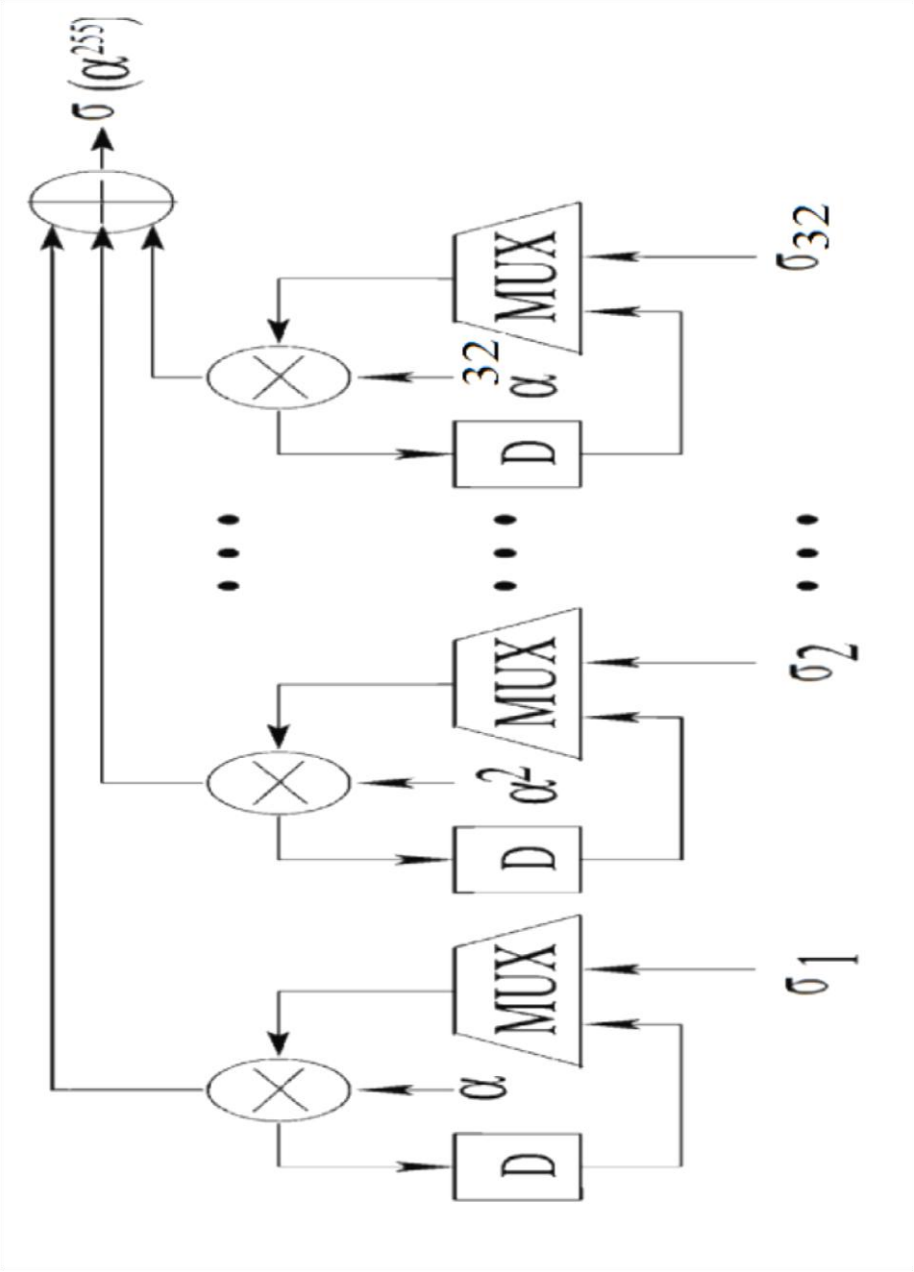



Figure 4.2 Chien's search architectures for (255, 223) Reed Solomon code

4- Evaluate error location numbers and error values and perform error correction.

In this step, the following polynomial is constructed:

$$Z(X) = 1 + (S_1 + \sigma_1)X + (S_2 + \sigma_1 S_1 + \sigma_2)X^2 + \ldots\ldots + (S_z + \sigma_1 S_{z-1} + \ldots\ldots + \sigma_{z-1} S_1 + \sigma_z)\, X^z$$

Then, the error value for the location given by $\beta_l = \alpha^{jl}$ is:

$$e_{j1} = \frac{Z(\beta_1^{\,n-1})}{\prod_{\substack{i=1 \\ i \neq 1}}^{z} (1 + \beta_i \beta_1^{\,n-1})}$$

The Chien search calculates the error value and adds it to the received codeword to make the correction. This step can be implemented in a straightforward form.

**Chapter 5**

**The hardware implementation encoder**

## 5.1.The hardware implementation encoder

Encoding RS- code can be done as follows. Let,

$m(X) = m_0 + m_1X + m_2X^2 + \ldots\ldots + m_{k-1}X^{k-1}$

be the message polynomial. In systematic form, the 2t parity check symbols are the coefficients of the remainder $b(X) = b_0 + b_1 X + \ldots\ldots + b_{2t-1}X^{2t-1}$ resulting from dividing the message polynomial $X^{2t}m(X)$ by the generator polynomial:

$b(X) + X^{2t}\, m(X) = a(X)\, g(X)$

Where:

g(X) is the generator polynomial

a(X) is the quotient from dividing the message polynomial by the generator polynomial

b(X) is the remainder

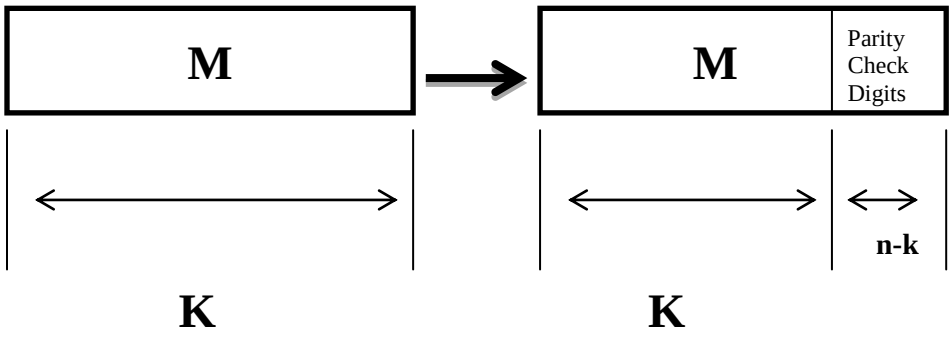



Figure 5.1 Systematic form

In hardware implementation for (255,223) R-S code, the generator polynomial g(X) is:

$$g(X) = 45+216X+239X^2+24X^3+253X^4+104X^5+27X^6+40X^7+107X^8+50X^9+163X^{10}+210X^{11}$$
$$+227X^{12}+134X^{13}+224X^{14}+158X^{15}+119X^{16}+13X^{17}+158X^{18}+X^{19}+238X^{20}+164X^{21}$$
$$+82X^{22}+43X^{23}+15X^{24}+232X^{25}+246X^{26}+142X^{27}+50X^{28}+189X^{29}+29X^{30}+232X^{31}+X^{32}$$

and the code polynomial v(X) of m(X) is:

$v(X) = b(X) + X^{32}\ m(X) = a(X)\ g(X)$

$v(X) = b_0 + b_1X + \ldots.+ b_{31}X^{31} + m_0X^{32}+ m_1X^{33}+\ldots.+ m_{222}X^{254}$

The encoder used for this Reed Solomon code has the following blocks:

- Thirty-two 8-bit wide adder that adds two field elements from GF($2^8$).
- Thirty-two multiplier that multiplies any field elements of G ($2^8$) by a fixed elements from same field.
- Thirty-two 8-bit registers to store a field element from the G($2^8$).
- One 8-bit wild multiplexer.
- One 8-bit wild AND gate.

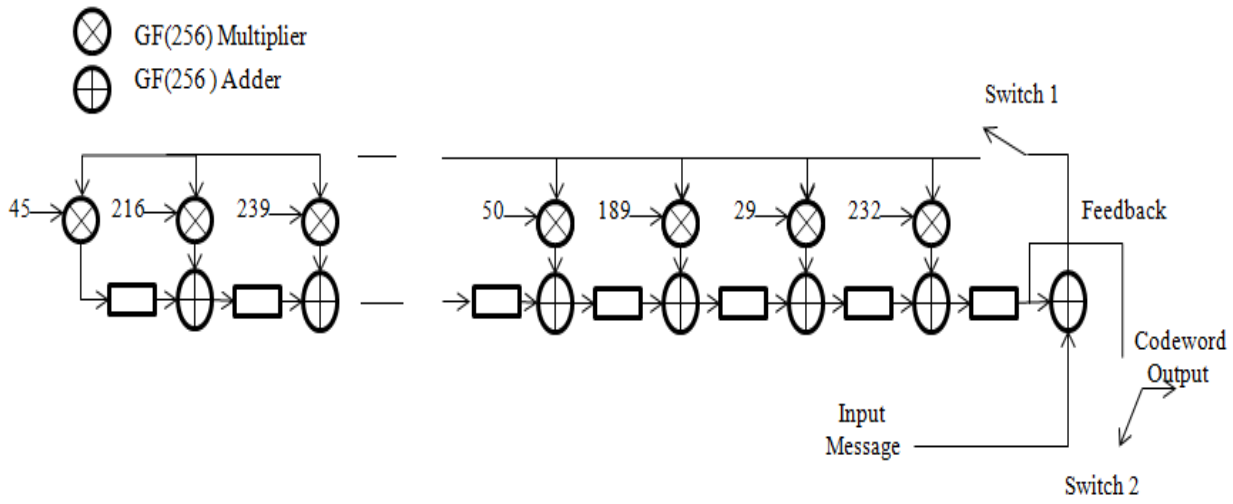


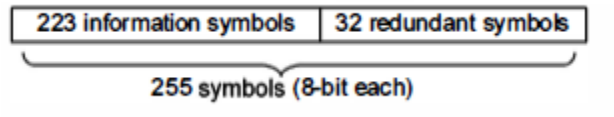

Figure 5.2 feedback shift register configuration Reed Solomon (255,223) encoder

The encoder operation is carried out as follows:

1- AND gate ON, so the 223 information digits are shifted into the shift register and simultaneously into the communication channel. As soon as the complete message has entered the shift register, the thirty-two digits in the register form the parity check digits.

2- AND gate OFF, so the feedback connection is disable.

The parity check digits are shifted out and sent into the channel so that the thirty two parity check digits together with the 223 information digits form a complete codeword. There will be 223 clock pulses to shift the information message into the shift register and the encoder output. Also, 32 clock-pulses shift the thirty-two parity check digits to the output, which means that there are 255 clock pulses in total.




Figuer 5.3 Codeword generated by Reed Solomon encoder

The encoder has three blocks, an input register, a processor register and an output register, as shown in Figure 5.4 . The encoder circuit is a single high-performance reduced instruction set computing RISC processor; as an example, 8-bit CMOS EPROM-ROM microcontroller PIC17C44 from Microchip Technology. This chip has a Harvard architecture as well as these features: a maximum frequency of operation of 33 MHz, the instruction cycle consists of four oscillator periods or the normal instruction execution time is 160 ns, and a transfer rate more than 100 Kbyte/sec on the communication channel .

The encoder transfers the input bytes of the message as well as computes the thirty two bytes parity check and, to have minimum bandwidth for the transmission channel, the output data rate has to be constant. As a result, the encoder is designed to be interrupt-driven and an external clock generates each interrupt. To speed up the system, the thirty two parity digits are implemented with one level ROM multiplication. For every multiplication, a table with 256 entries are calculated by multiplying any elements of GF($2^8$) with the coefficients of the generator polynomial g(X). The thirty two tables are stored in the program memory and then read through the table instructions of PIC17C44. The entries of the table are calculated in Matlab as well as C and shown in Appendix B. Figure 5.3 shows the flowcharts of the software implementation for the main module in the encoder, which only performs initializing the special purpose register, setting up the timer and performing the interrupt. Next, the processor will wait at a certain address until the external clock generates an interrupt. As the interrupt occurs, the processor will read the input then send the data byte to the output. After that, it will update all the thirty two parity check digits. The next step is to check if all the 223 bytes read and, if yes, shift out the thirty two parity bytes at the same rate as other bytes. If not, then return from the interrupt subroutine and wait for the next input.

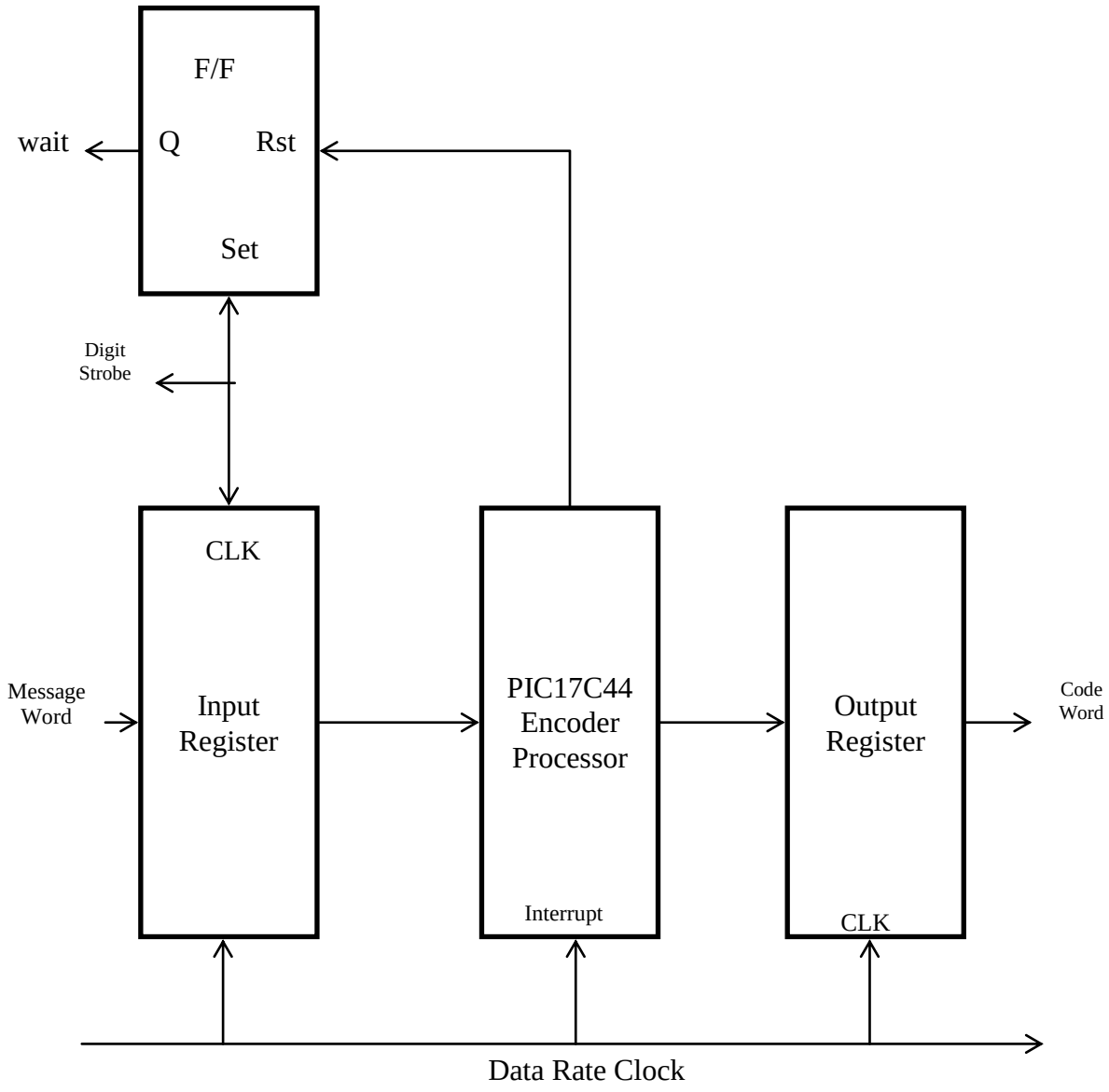



Figure 5.4 Encoder Block Diagram

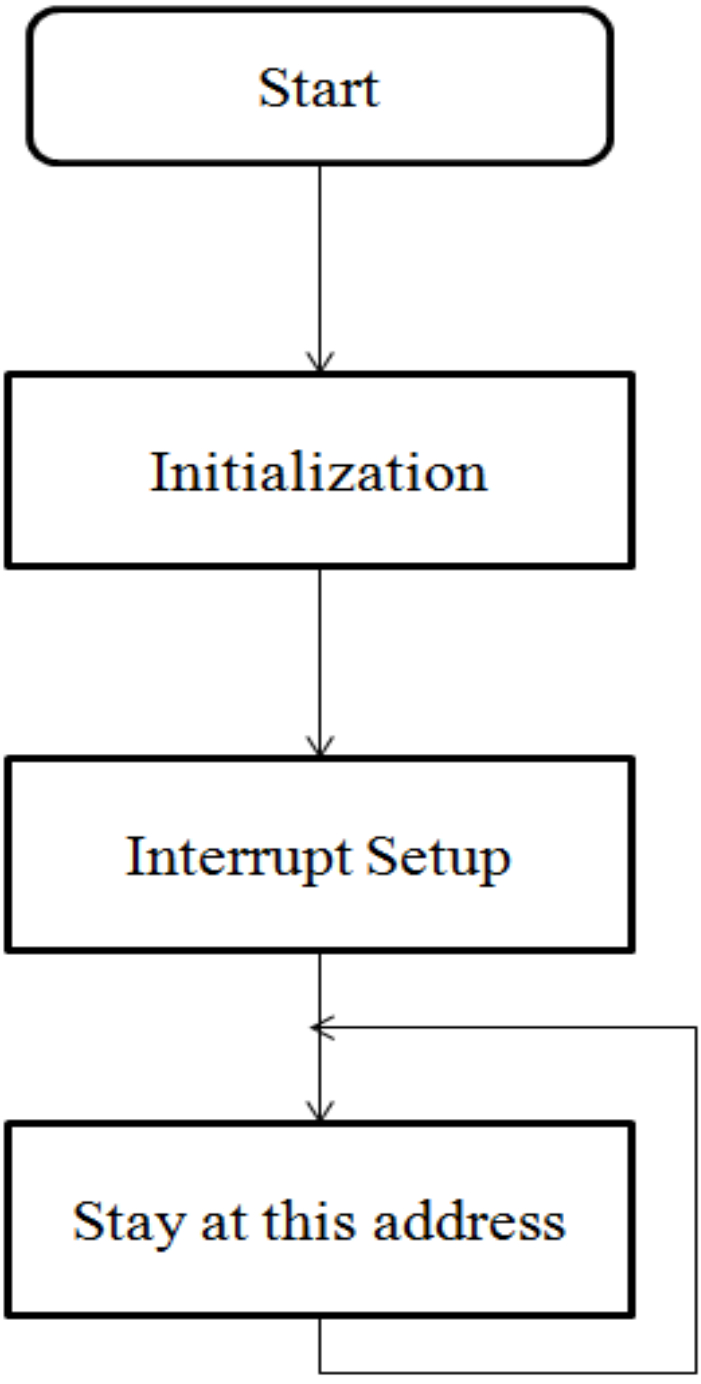



Figure 5.6 Encoder main module flowchart

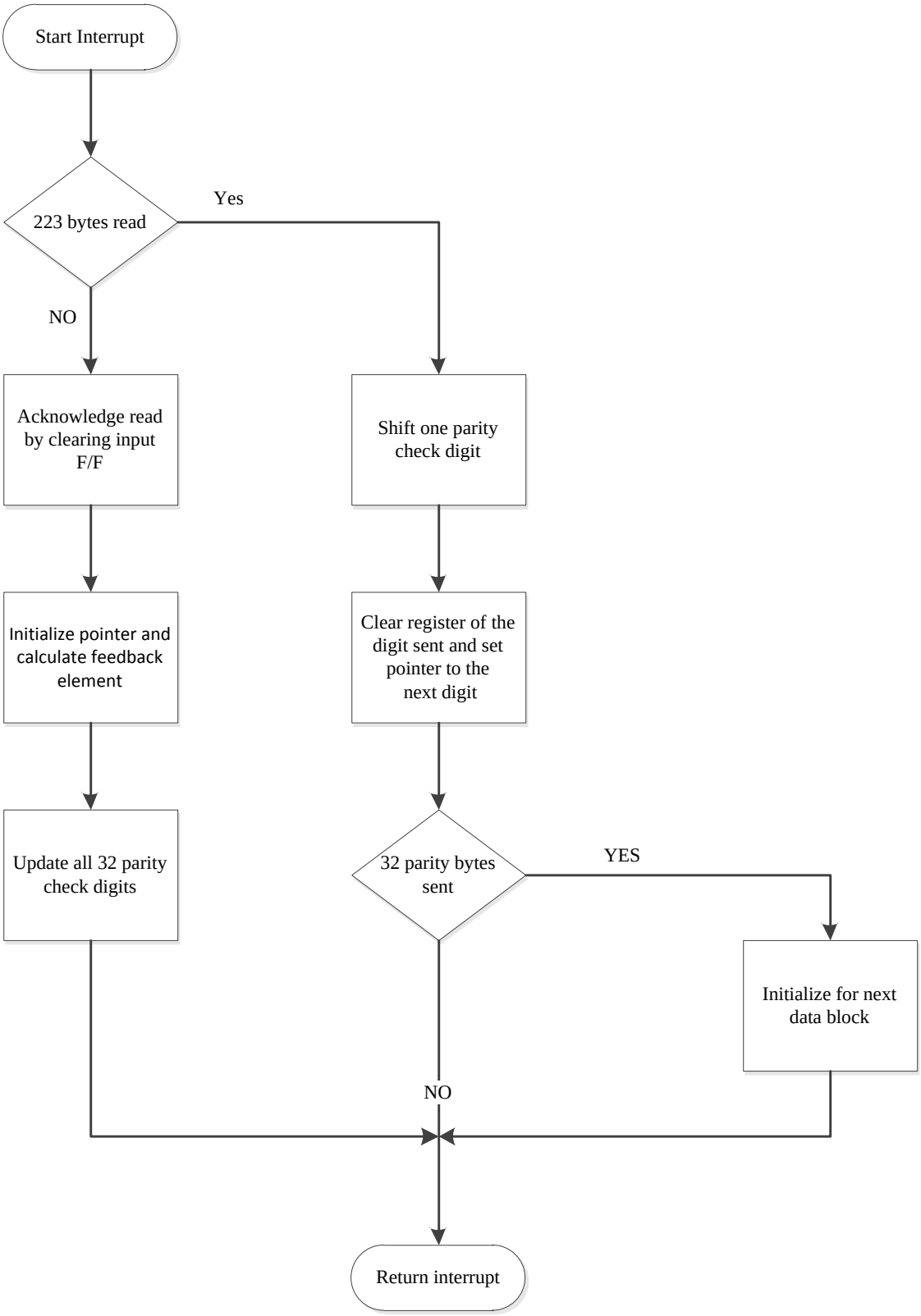



Figure 5.7 Encoder interrupt subroutine flowchart

# Chapter 6

# The hardware implementation decoder

## 6.1. The hardware implementation decoder

The decoder is more complex than the encoder due to the multiple stages of the decoding procedure outlined in chapter four.

The hardware implementation is done by using three single high-performance RISC microcontrollers. Three PIC17C44 microcontroller chips work in parallel for the decoder as shown in Figure 6.1.The first data block will be entered into the first microcontroller one byte at a time and saved in the internal RAM register. After the last byte is entered, the second microcontroller will take the next 255 bytes and then the third microcontroller will follow the same process. This process will give the first microcontroller the time to compute the error location, error location number, error value, and output the correct 223 byte message.

The syndrome computation is the stage in the decoder that will use more time as well as finding the roots of the error location polynomial. Therefore, sending out the entire syndrome in parallel will be the best. Since the data rate is uniform is using the channel effectively, it will give minimum bandwidth in the communication channel as well as a lower throughput. The flowcharts below show the decoding process outlined in chapter four. The code is presented in outline form to make it easier for the reader.

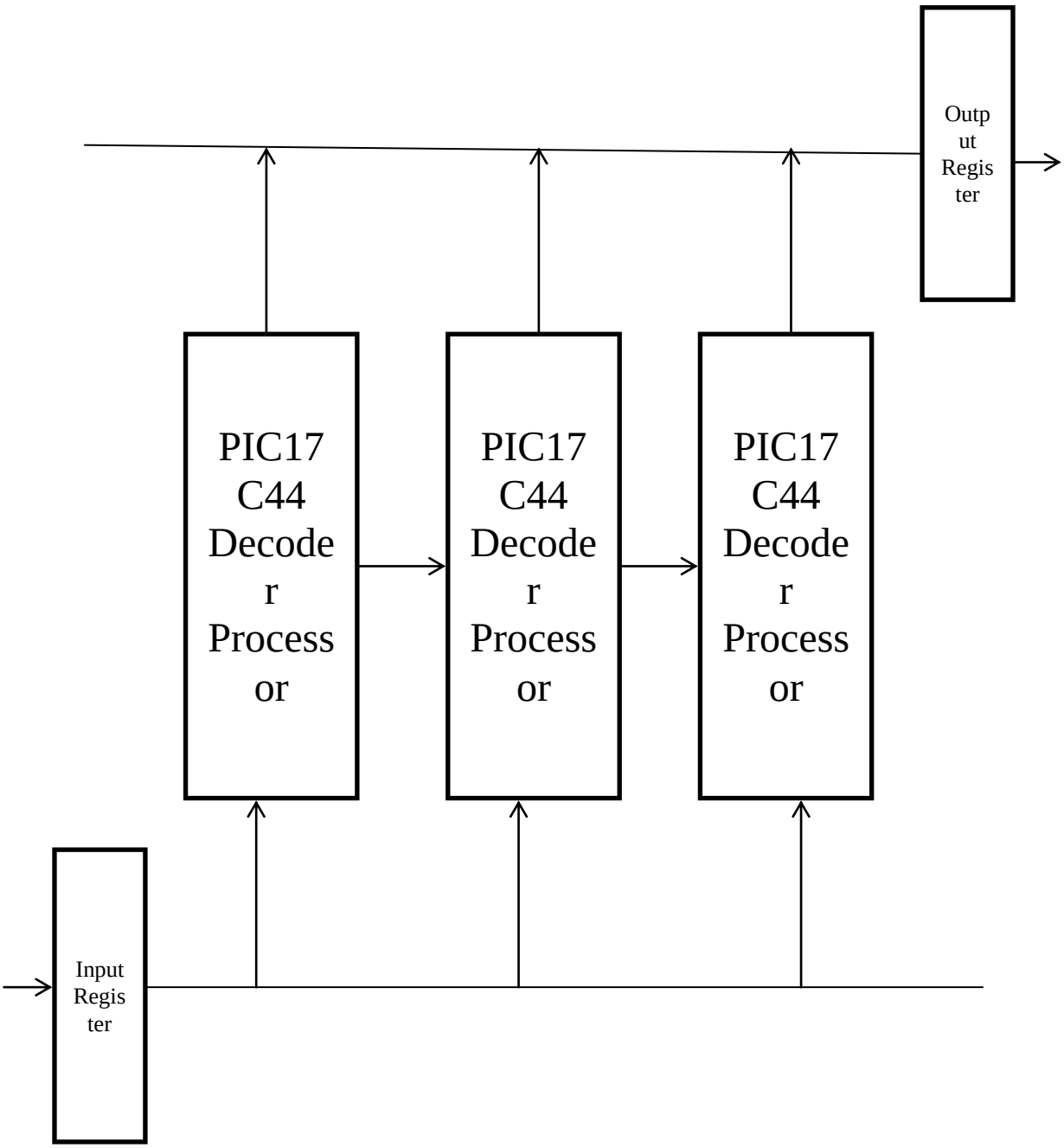



Figure 6.1 Decoder block diagram

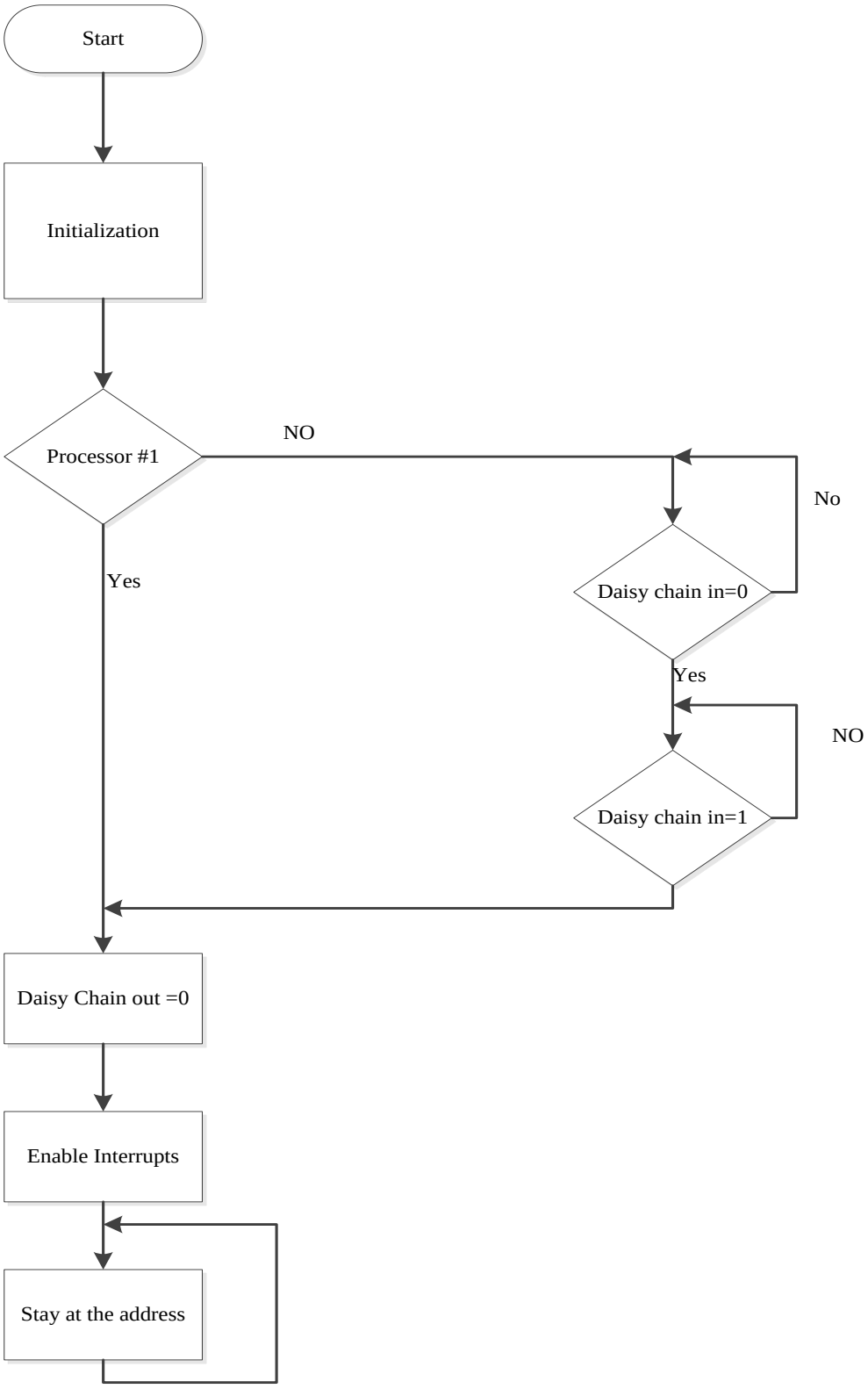



Figure 6.2 Decoder main module flowcharts

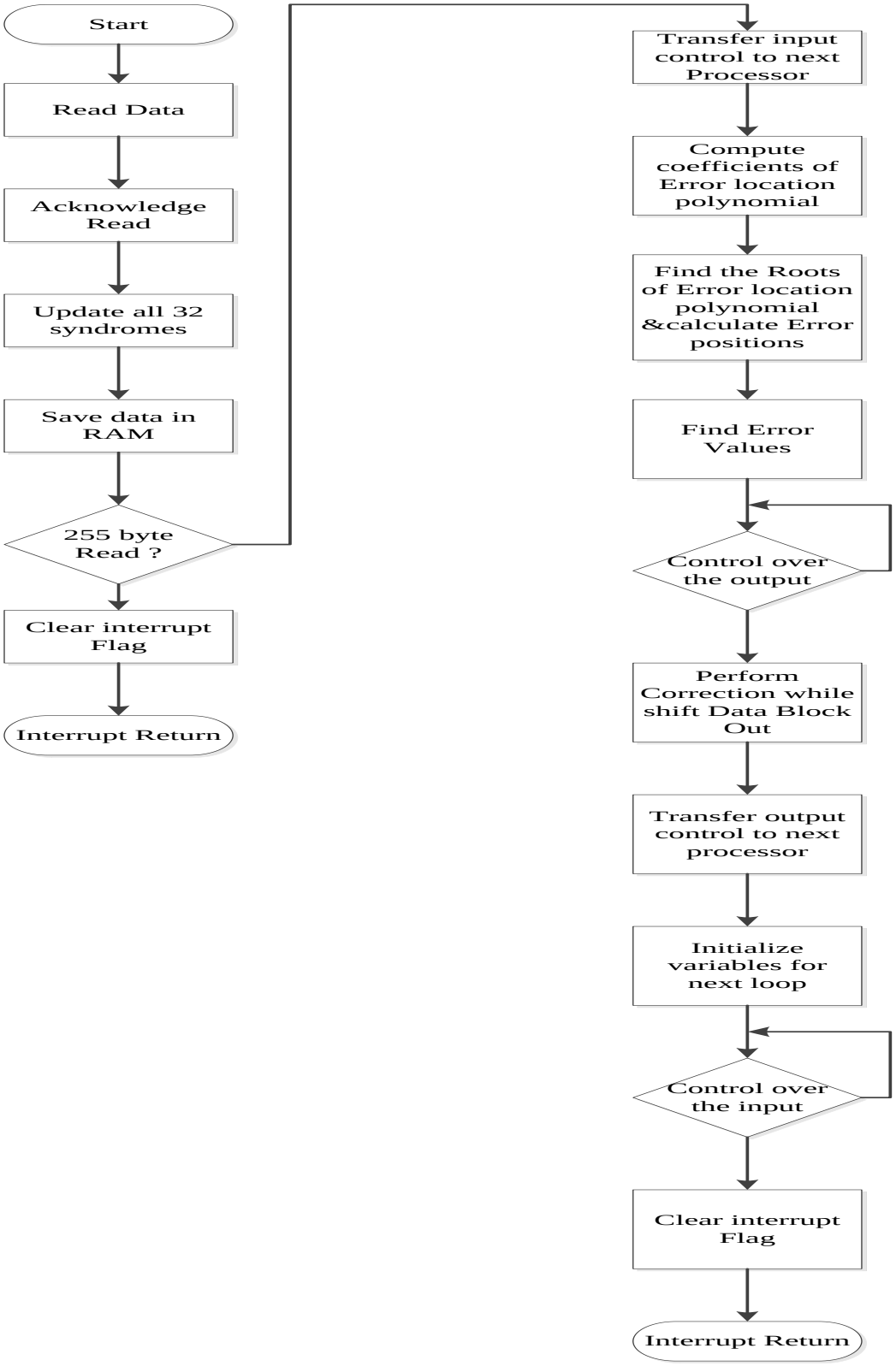



Figure 6.3 Decoder interrupt subroutine

# Chapter 7

# Testing the system

## 7.1.Testing the system

The (255, 223) R-S code project is not implemented in hardware, but is instead done with software on a PC Core 2Due CPU 2 GHz processor by compiling the written C source code with Eclipse. The code is then tested and provided good results as will be shown later in this chapter. Also, Simulink was used sometimes to confirm the same result as from the C file.

## 7.2.Test the code

### 7.2.1. Using Simulink

(255, 223) R-S code has been built in Simulink as shown in the Figure 7.1 with the following block:

1- Random Integer Generator, where M-ary number is set to 255, initial seed to a positive number, check the frame-based outputs check box and set samples per frame to 223.

2- Integer-Input RS Encoder, where codeword length N set to 255 and message length K to 223.

3- Gain block set to:

[0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; ones(223, 1)]

The 0's are to replace the first thirty two symbols in each codeword with zeros.

4- Integer-Output RS Decoder, where codeword length N is set to 255 and message length K to 223.

5- Set list of signs to |-+

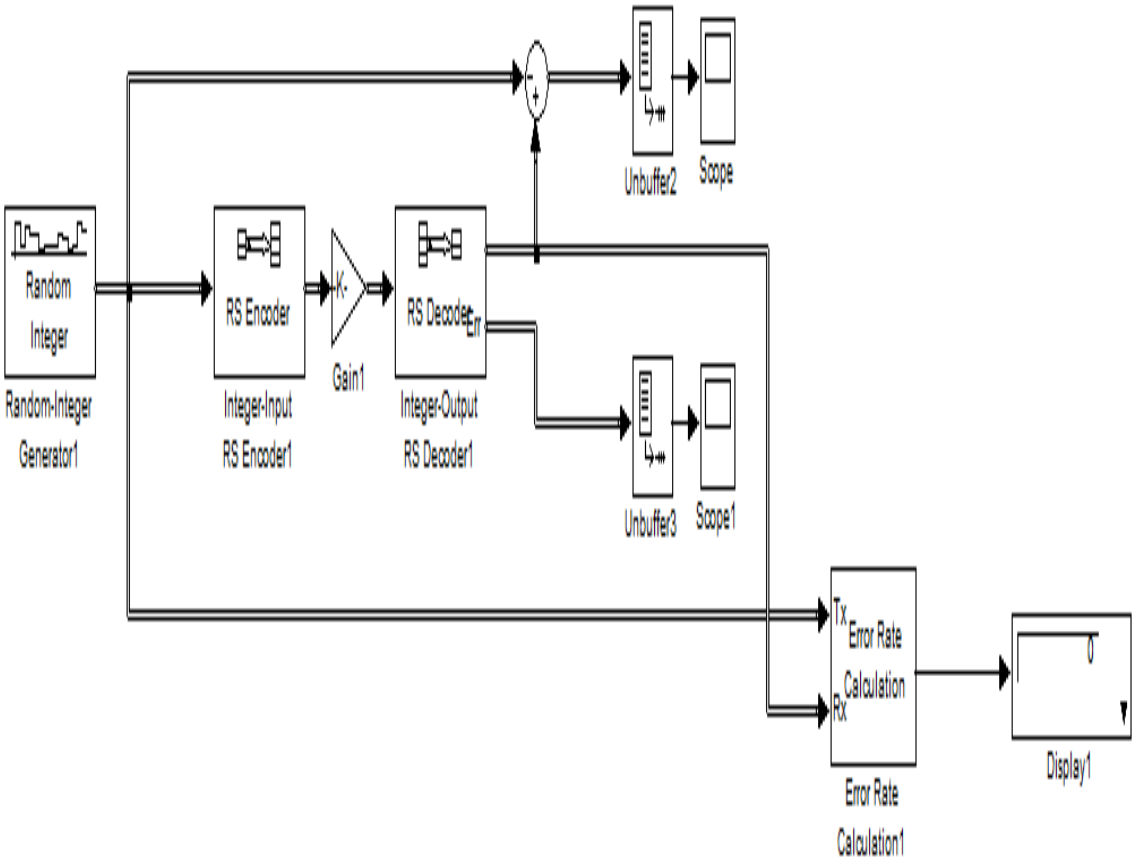



Figure 7.1 (255, 223) Reed Solomon Code in Simulink

After running the design first scope, Figure 7.2 gives the difference between the original message and the recovered message. Since the decoder was able to correct all errors that occurred, each of the sixteen data streams in the plot is zero as shown in Figure 7.2 and the error rate calculation between the input and the output message is zero as shown in Figure 7.3.

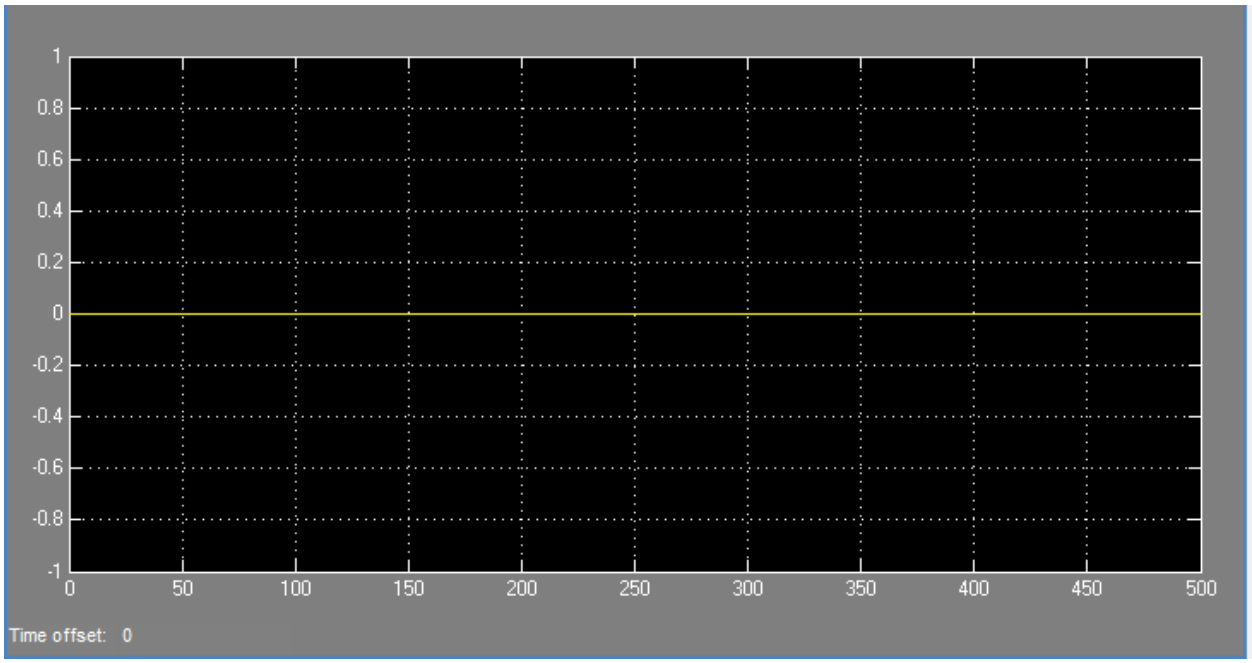



Figure 7.2 Output of scope 1

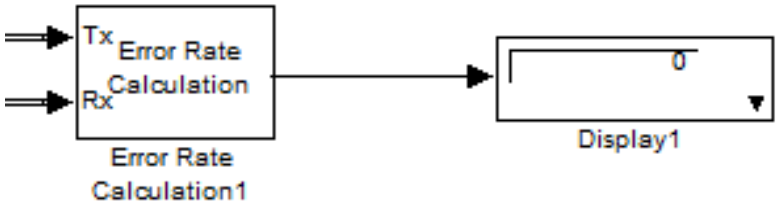



Figure 7.3 Error rate between the input and output message no error

The second plot Figure 7.4 gives the number of errors that the decoder detected while trying to recover the message. Often the number is thirty two because the gain replaces the thirty two first symbols in each codeword with zeros. However, the number of errors is less than thirty two whenever a correct codeword contains one or more zeros in the first thirty two places.

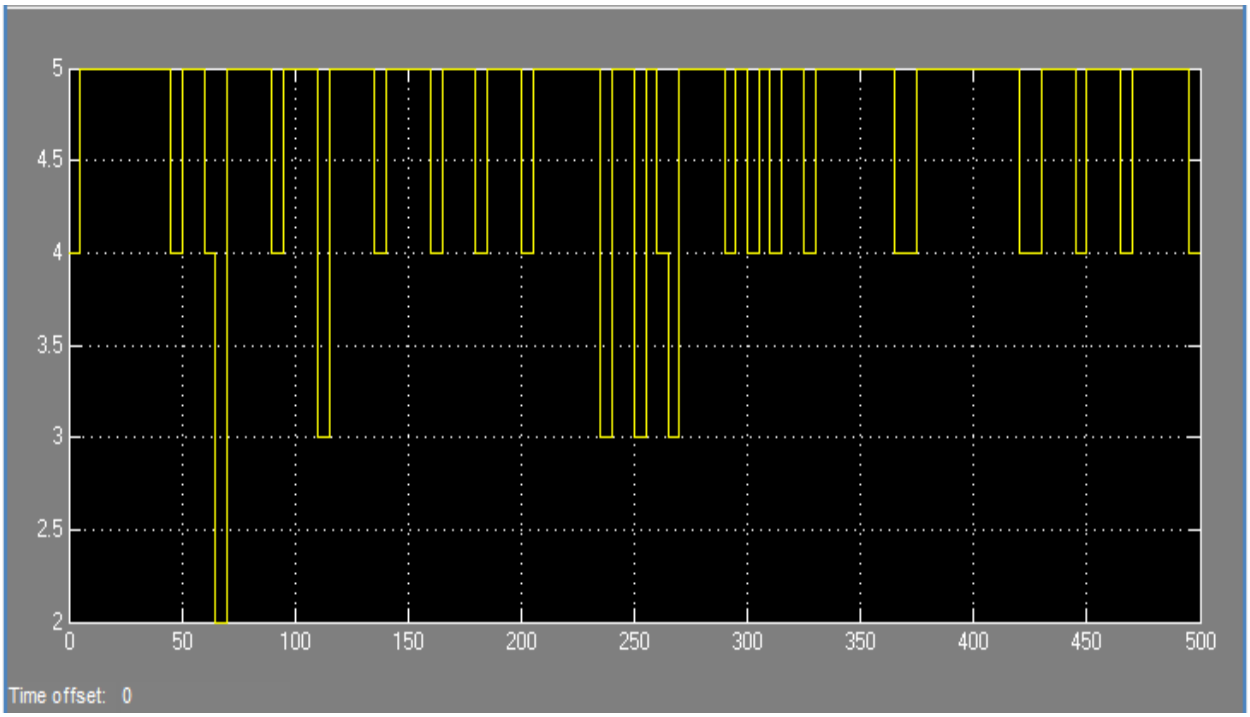



Figure 7.4 Output of scope 2

On the other hand, the decoder cannot correct all the errors in the codeword; this may be because the error received in the message happens to match a codeword. This will cause the decoder to clear as no error occur or the decoder cannot correct all the errors on the message. Figure 7.5 shows the error rate between the input and the output when the decoder is not able to correct all the errors.

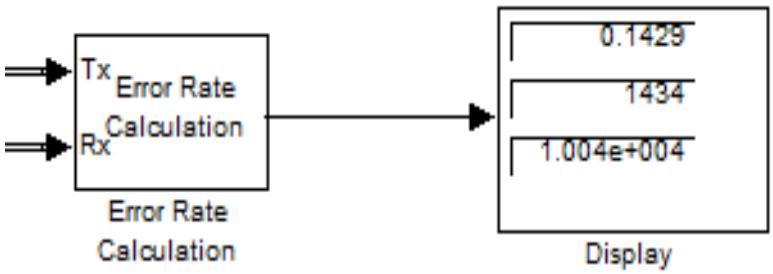



Figure 7.5 Error rate between the input and output message error occur

### 7.2.1. Using C program source code

This program is an encoder/decoder for (255, 223) Reed-Solomon codes. Encoding is in systematic form, decoding via the Berlekamp iterative algorithm. In the present form, the constants mm = 8, nn = 255, tt = 16, and kk = nn-2tt = 223 specified. Also, the irreducible polynomial used to generate GF (2**mm) has been entered, p(X) = $1+X^2+X^3+X^4+X^8$. The representation of the elements of GF (2**m) is either in index form, where the number is the power of the primitive element alpha, which is convenient for multiplication (add the powers modulo 2**m-1) or in polynomial form, where the bits represent the coefficients of the polynomial representation of the number, which is the most convenient form for addition. The two forms are swapped via lookup tables. This leads to fairly messy-looking expressions, but unfortunately, there is no easy alternative when working with Galois arithmetic. The code is not written in the most elegant way [9]. However, when including it into a simulation program, you may want to do some conversion of global variables to local variables where appropriate, and passing parameters, e.g. array addresses, to the functions may be a sensible strategy to reduce the number of global variables and thus decrease the chance of a bug being introduced.

The program has four main boxes: generator polynomial, encoder, decoder, and the main. However, the decoder by itself has four boxes as shown below where the first is finding the syndrome, the second is finding the error location polynomial, the third is finding the roots of the error location polynomial and finally the four is to evaluate errors. The output of each box is shown in Appendix B and the total time to run the software is 887ms.

Void generate_gf ( )

Generate GF(2**mm) from the irreducible polynomial p(X) in pp[0]..pp[mm]

Lookup tables: index->polynomial form alpha_to[] contains j=alpha**i;

polynomial form -> index form index_of [j=alpha**i] = i

alpha=2 is the primitive element of GF (2**mm)

```
#include <math.h>
#include <stdio.h>
#define mm 8                                    /* RS code over GF(2**8) - change to suit */
#define nn 255                                  /* nn=2**mm -1   length of codeword */
#define tt 16                                   /* number of errors that can be corrected */
#define kk 223                                  /* kk = nn-2*tt  */
int pp [mm+1] = { 1, 0, 1, 1, 1, 0, 0, 0, 1} ;  /* specify irreducible polynomial coeffts */
int alpha_to [nn+1], index_of [nn+1], gg [nn-kk+1] ;
int recd [nn], data [kk], bb [nn-kk];
{
  register int i, mask ;
  printf("void generate_gf()");
  mask = 1 ;
  alpha_to[mm] = 0 ;
  for (i=0; i<mm ; i++)
  {
        alpha_to[i] = mask ;
        index_of[alpha_to[i]] = i ;
 printf("\n  i=%3d  mask=%3d   mm=%3d  alpha_to[mm]=%3d  index_of[  alpha_to[i]  ]=%3d",
                  i,          mask,    mm,         alpha_to[mm],              index_of[alpha_to[i]] );
    if (pp[i]!=0)
      alpha_to[mm] ^= mask ;
    printf("\n  mask=%3d pp[%3d]=%3d   mm=%3d  alpha_to[mm]=%3d ",
                      mask,         i,   pp[i],          mm,               alpha_to[mm] );
    mask <<= 1 ;
  }
  index_of[alpha_to[mm]] = mm ;
  mask >>= 1 ;
  for (i=mm+1; i<nn; i++)
  {
         if (alpha_to[i-1] >= mask)
            alpha_to[i] = alpha_to[mm] ^ ((alpha_to[i-1]^mask)<<1) ;
        else alpha_to[i] = alpha_to[i-1]<<1 ;
            index_of[alpha_to[i]] = i ;
    printf("\nmask=%3d  i=%3d    alpha_to[i]=%3d    index_of[  alpha_to[i]   ]=%3d ",
                      mask,  i,              alpha_to[i],              index_of[alpha_to[i]] );
  }
 index_of[0] = -1 ;
}
```

Void gen_poly()

Obtain the generator polynomial of the tt-error correcting, length

nn=(2**mm -1) Reed Solomon code from the product of (X+alpha**i), i=1..2*tt

```
{
 register int i,j ;
 printf("\n  void gen_poly()");
 gg[0] = 2 ;                              /* primitive element alpha = 2  for GF(2**mm)  */
 gg[1] = 1 ;                             /* g(x) = (X+alpha) initially */
 for (i=0; i<=nn-kk; i++)
            printf("\n alpha_to[%2d]=%3d",i,alpha_to[i]);
 for (i=2; i<=nn-kk; i++)
     {
       gg[i] = 1 ;
 for (j=i-1; j>0; j--)
     if (gg[j] != 0)  gg[j] = gg[j-1]^ alpha_to[(index_of[gg[j]]+i)%nn] ;
     else gg[j] = gg[j-1] ;
   gg[0] = alpha_to[(index_of[gg[0]]+i)%nn] ;                    /* gg[0] can never be zero */
  }
/* convert gg[] to index form for quicker encoding */
      printf("\n  convert gg[] to index form for quicker encoding");
     for (i=0; i<=nn-kk; i++)
            printf("\nindex_of[%2d]=%3d",i,index_of[i]);
    for (i=0; i<=nn-kk; i++)
     {
            printf("\n gp gg[%2d ]=%3d  ", i, gg[i]);    /*display generator polynomial array */
            gg[i] = index_of[gg[i]] ;
     }
}
```

Void encode_rs()

Take the string of symbols in data[i], i=0…….(k-1) and encode systematically to produce 2*tt parity symbols in bb[0]…...bb[2*tt-1] data[] is input and bb[] is output in polynomial form. Encoding is done by using a feedback shift register with appropriate connections specified by the elements of gg[], which was generated above. Codeword is c(X) = data(X)*X**(nn-kk)+ b(X)

```
{
  register int i,j ;
  printf("\n  void encode_rs()");
  int feedback ;
  for (i=0; i<nn-kk; i++)     bb[i] = 0 ;
  for (i=kk-1; i>=0; i--)
      {
           feedback = index_of[data[i]^bb[nn-kk-1]] ;
     if (feedback != -1)
      {
          for (j=nn-kk-1; j>0; j--)
          if (gg[j] != -1)
          bb[j] = bb[j-1]^alpha_to[(gg[j]+feedback)%nn] ;
         else
           bb[j] = bb[j-1] ;
        bb[0] = alpha_to[(gg[0]+feedback)%nn] ;
      }
     else
      { for (j=nn-kk-1; j>0; j--)
         bb[j] = bb[j-1] ;
       bb[0] = 0 ;
      } ;
   } ;
} ;
```

Void decode_rs()

Assume we have received bits grouped into mm-bit symbols in recd[i], i=0..(nn-1), and recd[i] is index form (ie as powers of alpha). We first compute the 2*tt syndromes by substituting alpha**i into rec(X) and evaluating, storing the syndromes in s[i], i=1…….2tt (leave s[0] zero). Then we use the Berlekamp iteration to find the error location polynomial elp[i]. If the degree of the elp is >tt, we cannot correct all the errors and just put out the information symbols uncorrected. If the degree of elp is <=tt, we substitute alpha**i , i=1…….n into the elp to get the roots, the inverse roots, and the error location numbers. If the number of errors located does not equal the degree of the elp, we have more than tt errors and cannot correct them. Otherwise, we then solve for the error value at the error location and correct the error. For the cases where the number of errors is known to be too large to correct, the information symbols as received are output (the advantage of systematic encoding is that hopefully some of the information symbols will be okay and that the errors are in the parity part of the transmitted codeword). Of course, these insoluble cases can be returned as error flags to the calling routine if desired.

```
{
  register int i,j,u,q ;
  printf("\n  void decode_rs()");
  int elp[nn-kk+2][nn-kk], d[nn-kk+2], l[nn-kk+2], u_lu[nn-kk+2], s[nn-kk+1] ;
  int count=0, syn_error=0, root[tt], loc[tt], z[tt+1], err[nn], reg[tt+1] ;

/* first form the syndromes */
  for (i=1; i<=nn-kk; i++)
   { s[i] = 0 ;
     for (j=0; j<nn; j++)
       if (recd[j]!=-1)
         s[i] ^= alpha_to[(recd[j]+i*j)%nn] ;      /* recd[j] in index form */
/* convert syndrome from polynomial form to index form  */
     if (s[i]!=0)  syn_error=1 ;                   /* set flag if non-zero syndrome => error */
     s[i] = index_of[s[i]] ;
   } ;
  if (syn_error)                                   /* if errors, try and correct */
   {
```

Compute the error location polynomial via the Berlekamp iterative algorithm:

d[u] is the 'mu'th discrepancy, where u='mu'+1 and 'mu' (the Greek letter) is the step number ranging from -1 to 2*tt (see 4.3), l[u] is the degree of the elp at that step, and u_l[u] is the difference between the step number and the degree of the elp.

```
/* initialise table entries */
     d[0] = 0 ;            /* index form */
     d[1] = s[1] ;        /* index form */
     elp[0][0] = 0 ;      /* index form */
     elp[1][0] = 1 ;      /* polynomial form */
     for (i=1; i<nn-kk; i++)
       { elp[0][i] = -1 ;   /* index form */
         elp[1][i] = 0 ;   /* polynomial form */
       }
     l[0] = 0 ;
     l[1] = 0 ;
     u_lu[0] = -1 ;
     u_lu[1] = 0 ;
     u = 0 ;
     do
     {
       u++ ;
       if (d[u]==-1)
         { l[u+1] = l[u] ;
           for (i=0; i<=l[u]; i++)
            {  elp[u+1][i] = elp[u][i] ;
               elp[u][i] = index_of[elp[u][i]] ;
            }
         }
       else
/* search for words with greatest u_lu[q] for which d[q]!=0 */
         { q = u-1 ;
           while ((d[q]==-1) && (q>0)) q-- ;
/* have found first non-zero d[q]  */
           if (q>0)
            { j=q ;
              do
              { j-- ;
                if ((d[j]!=-1) && (u_lu[q]<u_lu[j]))
                   q = j ;
              }while (j>0) ;
            } ;
```

```
/* have now found q such that d[u]!=0 and u_lu[q] is maximum */
/* store degree of new elp polynomial */
          if (l[u]>l[q]+u-q)  l[u+1] = l[u] ;
          else  l[u+1] = l[q]+u-q ;
/* form new elp(x) */
          for (i=0; i<nn-kk; i++)    elp[u+1][i] = 0 ;
          for (i=0; i<=l[q]; i++)
            if (elp[q][i]!=-1)
              elp[u+1][i+u-q] = alpha_to[(d[u]+nn-d[q]+elp[q][i])%nn] ;
          for (i=0; i<=l[u]; i++)
            { elp[u+1][i] ^= elp[u][i] ;
              elp[u][i] = index_of[elp[u][i]] ;  /*convert old elp value to index*/
            }
        }
       u_lu[u+1] = u-l[u+1] ;
/* form (u+1)th discrepancy */
       if (u<nn-kk)    /* no discrepancy computed on last iteration */
         {
          if (s[u+1]!=-1)
                d[u+1] = alpha_to[s[u+1]] ;
          else
            d[u+1] = 0 ;
          for (i=1; i<=l[u+1]; i++)
            if ((s[u+1-i]!=-1) && (elp[u+1][i]!=0))
              d[u+1] ^= alpha_to[(s[u+1-i]+index_of[elp[u+1][i]])%nn] ;
          d[u+1] = index_of[d[u+1]] ;    /* put d[u+1] into index form */
         }
     } while ((u<nn-kk) && (l[u+1]<=tt)) ;
     u++ ;
     if (l[u]<=tt)         /* can correct error */
      {
/* put elp into index form */
        for (i=0; i<=l[u]; i++)   elp[u][i] = index_of[elp[u][i]] ;
```

```
/* find roots of the error location polynomial */
        for (i=1; i<=l[u]; i++)
          reg[i] = elp[u][i] ;
        count = 0 ;
        for (i=1; i<=nn; i++)
         {  q = 1 ;
           for (j=1; j<=l[u]; j++)
            if (reg[j]!=-1)
              { reg[j] = (reg[j]+j)%nn ;
                q ^= alpha_to[reg[j]] ;
              } ;
           if (!q)       /* store root and error location number indices */
            { root[count] = i;
              loc[count] = nn-i ;
              count++ ;
            };
         } ;
        if (count==l[u])    /* no. roots = degree of elp hence <= tt errors */
         {
/* form polynomial z(x) */
          for (i=1; i<=l[u]; i++)       /* Z[0] = 1 always - do not need */
           { if ((s[i]!=-1) && (elp[u][i]!=-1))
               z[i] = alpha_to[s[i]] ^ alpha_to[elp[u][i]] ;
             else if ((s[i]!=-1) && (elp[u][i]==-1))
                   z[i] = alpha_to[s[i]] ;
                 else if ((s[i]==-1) && (elp[u][i]!=-1))
                       z[i] = alpha_to[elp[u][i]] ;
                     else
                       z[i] = 0 ;
             for (j=1; j<i; j++)
               if ((s[j]!=-1) && (elp[u][i-j]!=-1))
                  z[i] ^= alpha_to[(elp[u][i-j] + s[j])%nn] ;
             z[i] = index_of[z[i]] ;         /* put into index form */
           } ;
```

```
/* evaluate errors at locations given by error location numbers loc[i] */
         for (i=0; i<nn; i++)
           { err[i] = 0 ;
             if (recd[i]!=-1)        /* convert recd[] to polynomial form */
               recd[i] = alpha_to[recd[i]] ;
             else  recd[i] = 0 ;
           }
         for (i=0; i<l[u]; i++)    /* compute numerator of error term first */
          { err[loc[i]] = 1;       /* accounts for z[0] */
            for (j=1; j<=l[u]; j++)
              if (z[j]!=-1)
                err[loc[i]] ^= alpha_to[(z[j]+j*root[i])%nn] ;
            if (err[loc[i]]!=0)
             { err[loc[i]] = index_of[err[loc[i]]] ;
               q = 0 ;     /* form denominator of error term */
               for (j=0; j<l[u]; j++)
                 if (j!=i)
                   q += index_of[1^alpha_to[(loc[j]+root[i])%nn]] ;
               q = q % nn ;
               err[loc[i]] = alpha_to[(err[loc[i]]-q+nn)%nn] ;
               recd[loc[i]] ^= err[loc[i]] ;  /*recd[i] must be in polynomial form */
             }
           }
         }
       else    /* no. roots != degree of elp => >tt errors and cannot solve */
         for (i=0; i<nn; i++)        /* could return error flag if desired */
            if (recd[i]!=-1)        /* convert recd[] to polynomial form */
              recd[i] = alpha_to[recd[i]] ;
            else  recd[i] = 0 ;     /* just output received codeword as is */
     }
    else         /* elp has degree has degree >tt hence cannot solve */
      for (i=0; i<nn; i++)       /* could return error flag if desired */
        if (recd[i]!=-1)        /* convert recd[] to polynomial form */
          recd[i] = alpha_to[recd[i]] ;
        else  recd[i] = 0 ;     /* just output received codeword as is */
   }
  else       /* no non-zero syndromes => no errors: output received codeword */
    for (i=0; i<nn; i++)
      if (recd[i]!=-1)        /* convert recd[] to polynomial form */
        recd[i] = alpha_to[recd[i]] ;
      else  recd[i] = 0 ;
 }
```

```
main()
{
  register int i;
/* generate the Galois Field GF(2**mm) */
  generate_gf() ;
  printf("\n Look-up tables for GF(2**%d)\n",mm) ;
  printf("  i   alpha_to[i]  index_of[i]\n") ;
  for (i=0; i<=nn; i++)
   printf("%3X      %3X          %3X\n",i,alpha_to[i],index_of[i]) ;
  printf("\n\n") ;
  for (i=0; i<=nn; i++)
   printf("%3d,",alpha_to[i]) ;
/* compute the generator polynomial for this RS code */
  gen_poly() ;
/* for known data, stick a few numbers into a zero codeword. Data is in polynomial form.*/
for  (i=0; i<kk; i++)   data[i] = kk - i ;
/* for example, say we transmit the following message (nothing special!) */
data[i] = kk - i
/* encode data[] to produce parity in bb[].  Data input and parity output is in polynomial
form*/
  encode_rs() ;
/* put the transmitted codeword, made up of data plus parity, in recd[] */
  for (i=0; i<nn-kk; i++)  recd[i] = bb[i] ;
  for (i=0; i<kk; i++) recd[i+nn-kk] = data[i] ;
/* if you want to test the program, corrupt some of the elements of recd[] here. This can also
be done easily in a debugger. */
/* Again, lets say that a middle element is changed */
  data[nn-nn/2] = 16 ;
  for( i=0; i<255; i++)
          printf("\nMessage    %3d   Output    %3d ", i, codeword[ i ] );
  for (i=0; i<nn; i++)
  {
          printf("\ni= %3d   recd[i]=%3d index_of[recd[i]]=%3d ", i, recd[i],
index_of[recd[i]] );
          recd[i] = index_of[recd[i]] ;          /* put recd[i] into index form */
          printf("\ni=%3d   recd[i]=%3d ", i, recd[i] );
  }
/* decode recv[] */
  decode_rs() ;         /* recd[] is returned in polynomial form */
/* print out the relevant stuff - initial and decoded {parity and message} */
  printf("\n Results for Reed-Solomon code (n=%3d, k=%3d, t= %3d)\n\n",nn,kk,tt) ;
  printf("  i  data[i]   recd[i](decoded)   (data, recd in polynomial form)\n");
  for (i=0; i<nn-kk; i++)
    printf("%3d    %3d      %3d\n",i, bb[i], recd[i]) ;
  for (i=nn-kk; i<nn; i++)
    printf("%3d    %3d      %3d\n",i, data[i-nn+kk], recd[i]) ;
}
```

# Chapter 8

# Conclusion

## 8.1. Conclusion

The communication channel in modern digital and data storage systems requires error detecting and correcting codes to correct the errors that occure during the transmission of data.

In this project, the algorithms of (255, 223) Reed Solomon code were first discussed and then designed and implemented using RISC microcontroller. The (255, 233) Reed Solomon code is defined over Galois Field GF ($2^8$) and is capable of correcting up to sixteen errors. A Reed Solomon code with different dimensions could have been chosen, but for the very common 8-bit data byte used today for many applications this code has been chosen.

The encoder is designed and implemented using linear feedback shift register (LFSR) and has been tested by simulation.

The decoder is more complicated than the encoder and, therefore, two level ROMs have been used and implemented in parallel to achieve a faster data rate. The decoder design is based on Berlekamp's iterative algorithm for finding the coefficients of the error location polynomial and Chien's searching algorithm for finding its roots.

The functionality of the encoder/decoder system was fully verified by simulation using Simulink and written in C language.

## References


[1] Lin, Shu. *An Introduction to Error Correcting Codes*. Englewood Cliffs, NJ: Prentice-Hall, 1970. Print.

[2] Reed, I. S., and G. Solomon. "Polynomial Codes Over Certain Finite Fields." *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960): 300. Print.

[3] Gallager, Robert G. *Information Theory and Reliable Communication,*. Wein: Springer-Verlag, 1972. Print.

[4] Sklar, Bernard. *Digital Communications: Fundamentals and Applications*. Upper Saddle River: Prentice Hall, 2001. Print.

[5] Odenwalder, Joseph P. *Error Control Coding Handbook Final Report*. San Diego, CA: Linkabit, 1976. Print.

[6] Berlekamp, E., R. Peile, and S. Pope. "The Application of Error Control to Communications." *IEEE Communications Magazine* 25.4 (1987): 44-57. Print.

[7] Hagenauer, J., and E. Lutz. "Forward Error Correction Coding for Fading Compensation in Mobile Satellite Channels." *IEEE Journal on Selected Areas in Communications* 5.2 (1987): 215-25. Print.

[8] Blahut, Richard E. *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley Pub., 1983. Print.

[9] Rockliff, Simon. "Encoder/Decoder for Reed-Solomon codes". 26 June 1991.1 September 2012 http://www.eccpage.com/rs.c

[10] microchip.com. "High-Performance 8-Bit CMOS EPROM/ROM Microcontroller". microchip.com 1 January 1998. 1 July 2012 http://ww1.microchip.com/downloads/en/DeviceDoc/30412c.pdf

[11] Wicker, Stephen B., and Vijay K. Bhargava. *Reed-Solomon Codes and Their Applications*. Piscataway, NJ: IEEE, 1994. Print.

[12] Mateeescu, Mihail V. *The Implementation Of a Reed Solomon Code Encoder/Decoder Using Microcontrollers*. Northridge: California State University,Northridge, 1998. Print.

[13] El Naga, Halima M. *Reed Solomon Code Encoder/Decoder Microprocessor Based System*. Northridge: California State University,Northridge, 1987. Print.

[14] Jia, Ong Jan. "Implementation of (255,223) Reed Solomon Minimal Instruction Set Computing Using Handel-C." *IEEE International Conference 9 (2010)*: 49-54. Print.

[15] Blahut, Richard E. *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley Pub., 1983. Print.

[17] Lin, Shu, and Daniel J. Costello. *Error Control Coding: Fundamentals and Applications*. Upper Saddle River, NJ: Pearson-Prentice Hall, 2004. Print.

[18] Roth, Ron. *Introduction to Coding Theory*. Cambridge: Cambridge UP, 2006. Print.

[19] Sklar, Bernard. *Digital Communications: Fundamentals and Applications*. Upper Saddle River: Prentice Hall, 2001. Print.

## Appendix A: Non zero elements of GF ($2^8$)

| Filed element in powers Of alpha | Filed element in DEC form | Filed element in HEX form | Filed element in BINARY Form "nonzero elements" |
|---|---|---|---|
| - | 0 | 00 | 0 0 0 0 0 0 0 0 |
| 0 | 1 | 01 | 1 0 0 0 0 0 0 0 |
| 1 | 2 | 02 | 0 1 0 0 0 0 0 0 |
| 2 | 4 | 04 | 0 0 1 0 0 0 0 0 |
| 3 | 8 | 08 | 0 0 0 1 0 0 0 0 |
| 4 | 16 | 10 | 0 0 0 0 1 0 0 0 |
| 5 | 32 | 20 | 0 0 0 0 0 1 0 0 |
| 6 | 64 | 40 | 0 0 0 0 0 0 1 0 |
| 7 | 128 | 80 | 0 0 0 0 0 0 0 1 |
| 8 | 29 | 1D | 1 0 1 1 1 0 0 0 |
| 9 | 58 | 3A | 0 1 0 1 1 1 0 0 |
| 10 | 116 | 74 | 0 0 1 0 1 1 1 0 |
| 11 | 232 | E8 | 0 0 0 1 0 1 1 1 |
| 12 | 205 | CD | 1 0 1 1 0 0 1 1 |
| 13 | 135 | 87 | 1 1 1 0 0 0 0 1 |
| 14 | 19 | 13 | 1 1 0 0 1 0 0 0 |
| 15 | 38 | 26 | 0 1 1 0 0 1 0 0 |
| 16 | 76 | 4C | 0 0 1 1 0 0 1 0 |
| 17 | 152 | 98 | 0 0 0 1 1 0 0 1 |
| 18 | 45 | 2D | 1 0 1 1 0 1 0 0 |
| 19 | 90 | 5A | 0 1 0 1 1 0 1 0 |
| 20 | 180 | B4 | 0 0 1 0 1 1 0 1 |
| 21 | 117 | 75 | 1 0 1 0 1 1 1 0 |
| 22 | 234 | EA | 0 1 0 1 0 1 1 1 |
| 23 | 201 | C9 | 1 0 0 1 0 0 1 1 |
| 24 | 143 | 8F | 1 1 1 1 0 0 0 1 |
| 25 | 3 | 3 | 1 1 0 0 0 0 0 0 |
| 26 | 6 | 6 | 0 1 1 0 0 0 0 0 |
| 27 | 12 | C | 0 0 1 1 0 0 0 0 |
| 28 | 24 | 18 | 0 0 0 1 1 0 0 0 |
| 29 | 48 | 30 | 0 0 0 0 1 1 0 0 |
| 30 | 96 | 60 | 0 0 0 0 0 1 1 0 |
| 31 | 192 | C0 | 0 0 0 0 0 0 1 1 |
| 32 | 157 | 9D | 1 0 1 1 1 0 0 1 |
| 33 | 39 | 27 | 1 1 1 0 0 1 0 0 |
| 34 | 78 | 4E | 0 1 1 1 0 0 1 0 |
| 35 | 156 | 9C | 0 0 1 1 1 0 0 1 |
| 36 | 37 | 25 | 1 0 1 0 0 1 0 0 |

| 37 | 74 | 4A | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 38 | 148 | 94 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 39 | 53 | 35 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 40 | 106 | 6A | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 41 | 212 | D4 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 42 | 181 | B5 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 43 | 119 | 77 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 44 | 238 | EE | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 45 | 193 | C1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 46 | 159 | 9F | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 47 | 35 | 23 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 48 | 70 | 46 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 49 | 140 | 8C | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 50 | 5 | 5 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 51 | 10 | A | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 52 | 20 | 14 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 53 | 40 | 28 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 54 | 80 | 50 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 55 | 160 | A0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 56 | 93 | 5D | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 57 | 186 | BA | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 58 | 105 | 69 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 59 | 210 | D2 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 60 | 185 | B9 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 61 | 111 | 6F | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 62 | 222 | DE | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 63 | 161 | A1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 64 | 95 | 5F | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 65 | 190 | BE | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 66 | 97 | 61 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 67 | 194 | C2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 68 | 153 | 99 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 69 | 47 | 2F | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 70 | 94 | 5E | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 71 | 188 | BC | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 72 | 101 | 65 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 73 | 202 | CA | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 74 | 137 | 89 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 75 | 15 | F | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 76 | 30 | 1E | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 77 | 60 | 3C | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 78 | 120 | 78 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 79 | 240 | F0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 80 | 253 | FD | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 81 | 231 | E7 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

| 82 | 211 | D3 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 83 | 187 | BB | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 84 | 107 | 6B | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 85 | 214 | D6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 86 | 177 | B1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 87 | 127 | 7F | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 88 | 254 | FE | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 89 | 225 | E1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 90 | 223 | DF | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 91 | 163 | A3 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 92 | 91 | 5B | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 93 | 182 | B6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 94 | 113 | 71 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 95 | 226 | E2 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 96 | 217 | D9 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 97 | 175 | AF | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 98 | 67 | 43 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 99 | 134 | 86 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 100 | 17 | 11 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 101 | 34 | 22 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 102 | 68 | 44 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 103 | 136 | 88 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 104 | 13 | D | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 105 | 26 | 1A | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 106 | 52 | 34 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 107 | 104 | 68 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 108 | 208 | D0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 109 | 189 | BD | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 110 | 103 | 67 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 111 | 206 | CE | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 112 | 129 | 81 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 113 | 31 | 1F | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 114 | 62 | 3E | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 115 | 124 | 7C | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 116 | 248 | F8 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 117 | 237 | ED | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 118 | 199 | C7 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 119 | 147 | 93 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 120 | 59 | 3B | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 121 | 118 | 76 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 122 | 236 | EC | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 123 | 197 | C5 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 124 | 151 | 97 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 125 | 51 | 33 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 126 | 102 | 66 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

| 127 | 204 | CC | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 128 | 133 | 85 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 129 | 23 | 17 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 130 | 46 | 2E | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 131 | 92 | 5C | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 132 | 184 | B8 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 133 | 109 | 6D | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 134 | 218 | DA | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 135 | 169 | A9 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 136 | 79 | 4F | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 137 | 158 | 9E | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 138 | 33 | 21 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 139 | 66 | 42 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 140 | 132 | 84 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 141 | 21 | 15 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 142 | 42 | 2A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 143 | 84 | 54 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 144 | 168 | A8 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 145 | 77 | 4D | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 146 | 154 | 9A | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 147 | 41 | 29 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 148 | 82 | 52 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 149 | 164 | A4 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 150 | 85 | 55 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 151 | 170 | AA | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 152 | 73 | 49 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 153 | 146 | 92 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 154 | 57 | 39 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 155 | 114 | 72 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 156 | 228 | E4 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 157 | 213 | D5 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 158 | 183 | B7 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 159 | 115 | 73 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 160 | 230 | E6 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 161 | 209 | D1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 162 | 191 | BF | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 163 | 99 | 63 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 164 | 198 | C6 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 165 | 145 | 91 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 166 | 63 | 3F | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 167 | 126 | 7E | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 168 | 252 | FC | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 169 | 229 | E5 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 170 | 215 | D7 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 171 | 179 | B3 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

| 172 | 123 | 7B | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 173 | 246 | F6 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 174 | 241 | F1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 175 | 255 | FF | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 176 | 227 | E3 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 177 | 219 | DB | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 178 | 171 | AB | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 179 | 75 | 4B | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 180 | 150 | 96 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 181 | 49 | 31 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 182 | 98 | 62 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 183 | 196 | C4 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 184 | 149 | 95 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 185 | 55 | 37 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 186 | 110 | 6E | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 187 | 220 | DC | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 188 | 165 | A5 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 189 | 87 | 57 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 190 | 174 | AE | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 191 | 65 | 41 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 192 | 130 | 82 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 193 | 25 | 19 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 194 | 50 | 32 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 195 | 100 | 64 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 196 | 200 | C8 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 197 | 141 | 8D | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 198 | 7 | 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 199 | 14 | E | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 200 | 28 | 1C | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 201 | 56 | 38 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 202 | 112 | 70 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 203 | 224 | E0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 204 | 221 | DD | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 205 | 167 | A7 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 206 | 83 | 53 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 207 | 166 | A6 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 208 | 81 | 51 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 209 | 162 | A2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 210 | 89 | 59 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 211 | 178 | B2 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 212 | 121 | 79 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 213 | 242 | F2 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 214 | 249 | F9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 215 | 239 | EF | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 216 | 195 | C3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

| 217 | 155 | 9B | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 218 | 43 | 2B | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 219 | 86 | 56 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 220 | 172 | AC | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 221 | 69 | 45 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 222 | 138 | 8A | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 223 | 9 | 9 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 224 | 18 | 12 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 225 | 36 | 24 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 226 | 72 | 48 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 227 | 144 | 90 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 228 | 61 | 3D | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 229 | 122 | 7A | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 230 | 244 | F4 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 231 | 245 | F5 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 232 | 247 | F7 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 233 | 243 | F3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 234 | 251 | FB | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 235 | 235 | EB | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 236 | 203 | CB | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 237 | 139 | 8B | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 238 | 11 | B | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 239 | 22 | 16 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 240 | 44 | 2C | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 241 | 88 | 58 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 242 | 176 | B0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 243 | 125 | 7D | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 244 | 250 | FA | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 245 | 233 | E9 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 246 | 207 | CF | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 247 | 131 | 83 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 248 | 27 | 1B | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 249 | 54 | 36 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 250 | 108 | 6C | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 251 | 216 | D8 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 252 | 173 | AD | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 253 | 71 | 47 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 254 | 142 | 8E | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Table A1. The field elements for GF (255) with $P(X) = 1+X^2+X^3+X^4+X^8$

## Appendix B: The ROMs tables

| Elem. GF (255) Of $\alpha^i$ | The ROMs Tables "HEX Form" Multiplying any elements from GF(255) with the relevant coefficients of the generator polynomial g(X) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $g_0 = \alpha^{18}$ | $g_1 = \alpha^{251}$ | $g_2 = \alpha^{215}$ | $g_3 = \alpha^{28}$ | $g_4 = \alpha^{80}$ | $g_5 = \alpha^{107}$ | $g_6 = \alpha^{248}$ | $g_7 = \alpha^{53}$ | $g_8 = \alpha^{84}$ | $g_9 = \alpha^{194}$ | $g_{10} = \alpha^{91}$ |
| - | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0 | 2D | D8 | EF | 18 | FD | 68 | 1B | 28 | 6B | 32 | A3 |
| 1 | 5A | AD | C3 | 30 | E7 | D0 | 36 | 50 | D6 | 64 | 5B |
| 2 | B4 | 47 | 9B | 60 | D3 | BD | 6C | A0 | B1 | C8 | B6 |
| 3 | 75 | 8E | 2B | C0 | BB | 67 | D8 | 5D | 7F | 8D | 71 |
| 4 | EA | 02 | 56 | 9D | 6B | CE | AD | BA | FE | 7 | E2 |
| 5 | C9 | 04 | AC | 27 | D6 | 81 | 47 | 69 | E1 | E | D9 |
| 6 | 8F | 08 | 45 | 4E | B1 | 1F | 8E | D2 | DF | 1C | AF |
| 7 | 3 | 10 | 8A | 9C | 7F | 3E | 02 | B9 | A3 | 38 | 43 |
| 8 | 6 | 20 | 9 | 25 | FE | 7C | 04 | 6F | 5B | 70 | 86 |
| 9 | C | 40 | 12 | 4A | E1 | F8 | 02 | DE | B6 | E0 | 11 |
| 10 | 18 | 80 | 24 | 94 | DF | ED | 04 | A1 | 71 | DD | 22 |
| 11 | 30 | 1D | 48 | 35 | A3 | C7 | 08 | 5F | E2 | A7 | 44 |
| 12 | 60 | 3A | 90 | 6A | 5B | 93 | 10 | BE | D9 | 53 | 88 |
| 13 | C0 | 74 | 3D | D4 | B6 | 3B | 20 | 61 | AF | A6 | D |
| 14 | 9D | E8 | 7A | B5 | 71 | 76 | 40 | C2 | 43 | 51 | 1A |
| 15 | 27 | CD | F4 | 77 | E2 | EC | 80 | 99 | 86 | A2 | 34 |
| 16 | 4E | 87 | F5 | EE | D9 | C5 | 1D | 2F | 11 | 59 | 68 |
| 17 | 9C | 13 | F7 | C1 | AF | 97 | 3A | 5E | 22 | B2 | D0 |
| 18 | 25 | 26 | F3 | 9F | 43 | 33 | 74 | BC | 44 | 79 | BD |
| 19 | 4A | 4C | FB | 23 | 86 | 66 | E8 | 65 | 88 | F2 | 67 |
| 20 | 94 | 98 | EB | 46 | 11 | CC | CD | CA | D | F9 | CE |
| 21 | 35 | 2D | CB | 8C | 22 | 85 | 87 | 89 | 1A | EF | 81 |
| 22 | 6A | 5A | 8B | 5 | 44 | 17 | 13 | F | 34 | C3 | 1F |
| 23 | D4 | B4 | B | A | 88 | 2E | 26 | 1E | 68 | 9B | 3E |
| 24 | B5 | 75 | 16 | 14 | D | 5C | 4C | 3C | D0 | 2B | 7C |
| 25 | 77 | EA | 2C | 28 | 1A | B8 | 98 | 78 | BD | 56 | F8 |
| 26 | EE | C9 | 58 | 50 | 34 | 6D | 2D | F0 | 67 | AC | ED |
| 27 | C1 | 8F | B0 | A0 | 68 | DA | 5A | FD | CE | 45 | C7 |
| 28 | 9F | 3 | 7D | 5D | D0 | A9 | B4 | E7 | 81 | 8A | 93 |
| 29 | 23 | 6 | FA | BA | BD | 4F | 75 | D3 | 1F | 9 | 3B |
| 30 | 46 | C | E9 | 69 | 67 | 9E | EA | BB | 3E | 12 | 76 |
| 31 | 8C | 18 | CF | D2 | CE | 21 | C9 | 6B | 7C | 24 | EC |
| 32 | 5 | 30 | 83 | B9 | 81 | 42 | 8F | D6 | F8 | 48 | C5 |
| 33 | A | 60 | 1B | 6F | 1F | 84 | 3 | B1 | ED | 90 | 97 |
| 34 | 14 | C0 | 36 | DE | 3E | 15 | 6 | 7F | C7 | 3D | 33 |
| 35 | 28 | 9D | 6C | A1 | 7C | 2A | C | FE | 93 | 7A | 66 |

| 36 | 50 | 27 | D8 | 5F | F8 | 54 | 18 | E1 | 3B | F4 | CC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 37 | A0 | 4E | AD | BE | ED | A8 | 30 | DF | 76 | F5 | 85 |
| 38 | 5D | 9C | 47 | 61 | C7 | 4D | 60 | A3 | EC | F7 | 17 |
| 39 | BA | 25 | 8E | C2 | 93 | 9A | C0 | 5B | C5 | F3 | 2E |
| 40 | 69 | 4A | 02 | 99 | 3B | 29 | 9D | B6 | 97 | FB | 5C |
| 41 | D2 | 94 | 04 | 2F | 76 | 52 | 27 | 71 | 33 | EB | B8 |
| 42 | B9 | 35 | 08 | 5E | EC | A4 | 4E | E2 | 66 | CB | 6D |
| 43 | 6F | 6A | 10 | BC | C5 | 55 | 9C | D9 | CC | 8B | DA |
| 44 | DE | D4 | 20 | 65 | 97 | AA | 25 | AF | 85 | B | A9 |
| 45 | A1 | B5 | 40 | CA | 33 | 49 | 4A | 43 | 17 | 16 | 4F |
| 46 | 5F | 77 | 80 | 89 | 66 | 92 | 94 | 86 | 2E | 2C | 9E |
| 47 | BE | EE | 1D | F | CC | 39 | 35 | 11 | 5C | 58 | 21 |
| 48 | 61 | C1 | 3A | 1E | 85 | 72 | 6A | 22 | B8 | B0 | 42 |
| 49 | C2 | 9F | 74 | 3C | 17 | E4 | D4 | 44 | 6D | 7D | 84 |
| 50 | 99 | 23 | E8 | 78 | 2E | D5 | B5 | 88 | DA | FA | 15 |
| 51 | 2F | 46 | CD | F0 | 5C | B7 | 77 | D | A9 | E9 | 2A |
| 52 | 5E | 8C | 87 | FD | B8 | 73 | EE | 1A | 4F | CF | 54 |
| 53 | BC | 5 | 13 | E7 | 6D | E6 | C1 | 34 | 9E | 83 | A8 |
| 54 | 65 | A | 26 | D3 | DA | D1 | 9F | 68 | 21 | 1B | 4D |
| 55 | CA | 14 | 4C | BB | A9 | BF | 23 | D0 | 42 | 36 | 9A |
| 56 | 89 | 28 | 98 | 6B | 4F | 63 | 46 | BD | 84 | 6C | 29 |
| 57 | F | 50 | 2D | D6 | 9E | C6 | 8C | 67 | 15 | D8 | 52 |
| 58 | 1E | A0 | 5A | B1 | 21 | 91 | 5 | CE | 2A | AD | A4 |
| 59 | 3C | 5D | B4 | 7F | 42 | 3F | A | 81 | 54 | 47 | 55 |
| 60 | 78 | BA | 75 | FE | 84 | 7E | 14 | 1F | A8 | 8E | AA |
| 61 | F0 | 69 | EA | E1 | 15 | FC | 28 | 3E | 4D | 02 | 49 |
| 62 | FD | D2 | C9 | DF | 2A | E5 | 50 | 7C | 9A | 04 | 92 |
| 63 | E7 | B9 | 8F | A3 | 54 | D7 | A0 | F8 | 29 | 08 | 39 |
| 64 | D3 | 6F | 3 | 5B | A8 | B3 | 5D | ED | 52 | 10 | 72 |
| 65 | BB | DE | 6 | B6 | 4D | 7B | BA | C7 | A4 | 20 | E4 |
| 66 | 6B | A1 | C | 71 | 9A | F6 | 69 | 93 | 55 | 40 | D5 |
| 67 | D6 | 5F | 18 | E2 | 29 | F1 | D2 | 3B | AA | 80 | B7 |
| 68 | B1 | BE | 30 | D9 | 52 | FF | B9 | 76 | 49 | 1D | 73 |
| 69 | 7F | 61 | 60 | AF | A4 | E3 | 6F | EC | 92 | 3A | E6 |
| 70 | FE | C2 | C0 | 43 | 55 | DB | DE | C5 | 39 | 74 | D1 |
| 71 | E1 | 99 | 9D | 86 | AA | AB | A1 | 97 | 72 | E8 | BF |
| 72 | DF | 2F | 27 | 11 | 49 | 4B | 5F | 33 | E4 | CD | 63 |
| 73 | A3 | 5E | 4E | 22 | 92 | 96 | BE | 66 | D5 | 87 | C6 |
| 74 | 5B | BC | 9C | 44 | 39 | 31 | 61 | CC | B7 | 13 | 91 |
| 75 | B6 | 65 | 25 | 88 | 72 | 62 | C2 | 85 | 73 | 26 | 3F |
| 76 | 71 | CA | 4A | D | E4 | C4 | 99 | 17 | E6 | 4C | 7E |
| 77 | E2 | 89 | 94 | 1A | D5 | 95 | 2F | 2E | D1 | 98 | FC |
| 78 | D9 | F | 35 | 34 | B7 | 37 | 5E | 5C | BF | 2D | E5 |
| 79 | AF | 1E | 6A | 68 | 73 | 6E | BC | B8 | 63 | 5A | D7 |
| 80 | 43 | 3C | D4 | D0 | E6 | DC | 65 | 6D | C6 | B4 | B3 |

| 81 | 86 | 78 | B5 | BD | D1 | A5 | CA | DA | 91 | 75 | 7B |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 82 | 11 | F0 | 77 | 67 | BF | 57 | 89 | A9 | 3F | EA | F6 |
| 83 | 22 | FD | EE | CE | 63 | AE | F | 4F | 7E | C9 | F1 |
| 84 | 44 | E7 | C1 | 81 | C6 | 41 | 1E | 9E | FC | 8F | FF |
| 85 | 88 | D3 | 9F | 1F | 91 | 82 | 3C | 21 | E5 | 3 | E3 |
| 86 | D | BB | 23 | 3E | 3F | 19 | 78 | 42 | D7 | 6 | DB |
| 87 | 1A | 6B | 46 | 7C | 7E | 32 | F0 | 84 | B3 | C | AB |
| 88 | 34 | D6 | 8C | F8 | FC | 64 | FD | 15 | 7B | 18 | 4B |
| 89 | 68 | B1 | 5 | ED | E5 | C8 | E7 | 2A | F6 | 30 | 96 |
| 90 | D0 | 7F | A | C7 | D7 | 8D | D3 | 54 | F1 | 60 | 31 |
| 91 | BD | FE | 14 | 93 | B3 | 7 | BB | A8 | FF | C0 | 62 |
| 92 | 67 | E1 | 28 | 3B | 7B | E | 6B | 4D | E3 | 9D | C4 |
| 93 | CE | DF | 50 | 76 | F6 | 1C | D6 | 9A | DB | 27 | 95 |
| 94 | 81 | A3 | A0 | EC | F1 | 38 | B1 | 29 | AB | 4E | 37 |
| 95 | 1F | 5B | 5D | C5 | FF | 70 | 7F | 52 | 4B | 9C | 6E |
| 96 | 3E | B6 | BA | 97 | E3 | E0 | FE | A4 | 96 | 25 | DC |
| 97 | 7C | 71 | 69 | 33 | DB | DD | E1 | 55 | 31 | 4A | A5 |
| 98 | F8 | E2 | D2 | 66 | AB | A7 | DF | AA | 62 | 94 | 57 |
| 99 | ED | D9 | B9 | CC | 4B | 53 | A3 | 49 | C4 | 35 | AE |
| 100 | C7 | AF | 6F | 85 | 96 | A6 | 5B | 92 | 95 | 6A | 41 |
| 101 | 93 | 43 | DE | 17 | 31 | 51 | B6 | 39 | 37 | D4 | 82 |
| 102 | 3B | 86 | A1 | 2E | 62 | A2 | 71 | 72 | 6E | B5 | 19 |
| 103 | 76 | 11 | 5F | 5C | C4 | 59 | E2 | E4 | DC | 77 | 32 |
| 104 | EC | 22 | BE | B8 | 95 | B2 | D9 | D5 | A5 | EE | 64 |
| 105 | C5 | 44 | 61 | 6D | 37 | 79 | AF | B7 | 57 | C1 | C8 |
| 106 | 97 | 88 | C2 | DA | 6E | F2 | 43 | 73 | AE | 9F | 8D |
| 107 | 33 | D | 99 | A9 | DC | F9 | 86 | E6 | 41 | 23 | 7 |
| 108 | 66 | 1A | 2F | 4F | A5 | EF | 11 | D1 | 82 | 46 | E |
| 109 | CC | 34 | 5E | 9E | 57 | C3 | 22 | BF | 19 | 8C | 1C |
| 110 | 85 | 68 | BC | 21 | AE | 9B | 44 | 63 | 32 | 5 | 38 |
| 111 | 17 | D0 | 65 | 42 | 41 | 2B | 88 | C6 | 64 | A | 70 |
| 112 | 2E | BD | CA | 84 | 82 | 56 | D | 91 | C8 | 14 | E0 |
| 113 | 5C | 67 | 89 | 15 | 19 | AC | 1A | 3F | 8D | 28 | DD |
| 114 | B8 | CE | F | 2A | 32 | 45 | 34 | 7E | 7 | 50 | A7 |
| 115 | 6D | 81 | 1E | 54 | 64 | 8A | 68 | FC | E | A0 | 53 |
| 116 | DA | 1F | 3C | A8 | C8 | 9 | D0 | E5 | 1C | 5D | A6 |
| 117 | A9 | 3E | 78 | 4D | 8D | 12 | BD | D7 | 38 | BA | 51 |
| 118 | 4F | 7C | F0 | 9A | 7 | 24 | 67 | B3 | 70 | 69 | A2 |
| 119 | 9E | F8 | FD | 29 | E | 48 | CE | 7B | E0 | D2 | 59 |
| 120 | 21 | ED | E7 | 52 | 1C | 90 | 81 | F6 | DD | B9 | B2 |
| 121 | 42 | C7 | D3 | A4 | 38 | 3D | 1F | F1 | A7 | 6F | 79 |
| 122 | 84 | 93 | BB | 55 | 70 | 7A | 3E | FF | 53 | DE | F2 |
| 123 | 15 | 3B | 6B | AA | E0 | F4 | 7C | E3 | A6 | A1 | F9 |
| 124 | 2A | 76 | D6 | 49 | DD | F5 | F8 | DB | 51 | 5F | EF |
| 125 | 54 | EC | B1 | 92 | A7 | F7 | ED | AB | A2 | BE | C3 |

| 126 | A8 | C5 | 7F | 39 | 53 | F3 | C7 | 4B | 59 | 61 | 9B |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 127 | 4D | 97 | FE | 72 | A6 | FB | 93 | 96 | B2 | C2 | 2B |
| 128 | 9A | 33 | E1 | E4 | 51 | EB | 3B | 31 | 79 | 99 | 56 |
| 129 | 29 | 66 | DF | D5 | A2 | CB | 76 | 62 | F2 | 2F | AC |
| 130 | 52 | CC | A3 | B7 | 59 | 8B | EC | C4 | F9 | 5E | 45 |
| 131 | A4 | 85 | 5B | 73 | B2 | B | C5 | 95 | EF | BC | 8A |
| 132 | 55 | 17 | B6 | E6 | 79 | 16 | 97 | 37 | C3 | 65 | 9 |
| 133 | AA | 2E | 71 | D1 | F2 | 2C | 33 | 6E | 9B | CA | 12 |
| 134 | 49 | 5C | E2 | BF | F9 | 58 | 66 | DC | 2B | 89 | 24 |
| 135 | 92 | B8 | D9 | 63 | EF | B0 | CC | A5 | 56 | F | 48 |
| 136 | 39 | 6D | AF | C6 | C3 | 7D | 85 | 57 | AC | 1E | 90 |
| 137 | 72 | DA | 43 | 91 | 9B | FA | 17 | AE | 45 | 3C | 3D |
| 138 | E4 | A9 | 86 | 3F | 2B | E9 | 2E | 41 | 8A | 78 | 7A |
| 139 | D5 | 4F | 11 | 7E | 56 | CF | 5C | 82 | 9 | F0 | F4 |
| 140 | B7 | 9E | 22 | FC | AC | 83 | B8 | 19 | 12 | FD | F5 |
| 141 | 73 | 21 | 44 | E5 | 45 | 1B | 6D | 32 | 24 | E7 | F7 |
| 142 | E6 | 42 | 88 | D7 | 8A | 36 | DA | 64 | 48 | D3 | F3 |
| 143 | D1 | 84 | D | B3 | 9 | 6C | A9 | C8 | 90 | BB | FB |
| 144 | BF | 15 | 1A | 7B | 12 | D8 | 4F | 8D | 3D | 6B | EB |
| 145 | 63 | 2A | 34 | F6 | 24 | AD | 9E | 7 | 7A | D6 | CB |
| 146 | C6 | 54 | 68 | F1 | 48 | 47 | 21 | E | F4 | B1 | 8B |
| 147 | 91 | A8 | D0 | FF | 90 | 8E | 42 | 1C | F5 | 7F | B |
| 148 | 3F | 4D | BD | E3 | 3D | 02 | 84 | 38 | F7 | FE | 16 |
| 149 | 7E | 9A | 67 | DB | 7A | 04 | 15 | 70 | F3 | E1 | 2C |
| 150 | FC | 29 | CE | AB | F4 | 08 | 2A | E0 | FB | DF | 58 |
| 151 | E5 | 52 | 81 | 4B | F5 | 10 | 54 | DD | EB | A3 | B0 |
| 152 | D7 | A4 | 1F | 96 | F7 | 20 | A8 | A7 | CB | 5B | 7D |
| 153 | B3 | 55 | 3E | 31 | F3 | 40 | 4D | 53 | 8B | B6 | FA |
| 154 | 7B | AA | 7C | 62 | FB | 80 | 9A | A6 | B | 71 | E9 |
| 155 | F6 | 49 | F8 | C4 | EB | 1D | 29 | 51 | 16 | E2 | CF |
| 156 | F1 | 92 | ED | 95 | CB | 3A | 52 | A2 | 2C | D9 | 83 |
| 157 | FF | 39 | C7 | 37 | 8B | 74 | A4 | 59 | 58 | AF | 1B |
| 158 | E3 | 72 | 93 | 6E | B | E8 | 55 | B2 | B0 | 43 | 36 |
| 159 | DB | E4 | 3B | DC | 16 | CD | AA | 79 | 7D | 86 | 6C |
| 160 | AB | D5 | 76 | A5 | 2C | 87 | 49 | F2 | FA | 11 | D8 |
| 161 | 4B | B7 | EC | 57 | 58 | 13 | 92 | F9 | E9 | 22 | AD |
| 162 | 96 | 73 | C5 | AE | B0 | 26 | 39 | EF | CF | 44 | 47 |
| 163 | 31 | E6 | 97 | 41 | 7D | 4C | 72 | C3 | 83 | 88 | 8E |
| 164 | 62 | D1 | 33 | 82 | FA | 98 | E4 | 9B | 1B | D | 02 |
| 165 | C4 | BF | 66 | 19 | E9 | 2D | D5 | 2B | 36 | 1A | 04 |
| 166 | 95 | 63 | CC | 32 | CF | 5A | B7 | 56 | 6C | 34 | 08 |
| 167 | 37 | C6 | 85 | 64 | 83 | B4 | 73 | AC | D8 | 68 | 10 |
| 168 | 6E | 91 | 17 | C8 | 1B | 75 | E6 | 45 | AD | D0 | 20 |
| 169 | DC | 3F | 2E | 8D | 36 | EA | D1 | 8A | 47 | BD | 40 |
| 170 | A5 | 7E | 5C | 7 | 6C | C9 | BF | 9 | 8E | 67 | 80 |

| 171 | 57 | FC | B8 | E | D8 | 8F | 63 | 12 | 02 | CE | 1D |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 172 | AE | E5 | 6D | 1C | AD | 3 | C6 | 24 | 04 | 81 | 3A |
| 173 | 41 | D7 | DA | 38 | 47 | 6 | 91 | 48 | 08 | 1F | 74 |
| 174 | 82 | B3 | A9 | 70 | 8E | C | 3F | 90 | 10 | 3E | E8 |
| 175 | 19 | 7B | 4F | E0 | 02 | 18 | 7E | 3D | 20 | 7C | CD |
| 176 | 32 | F6 | 9E | DD | 04 | 30 | FC | 7A | 40 | F8 | 87 |
| 177 | 64 | F1 | 21 | A7 | 08 | 60 | E5 | F4 | 80 | ED | 13 |
| 178 | C8 | FF | 42 | 53 | 10 | C0 | D7 | F5 | 1D | C7 | 26 |
| 179 | 8D | E3 | 84 | A6 | 20 | 9D | B3 | F7 | 3A | 93 | 4C |
| 180 | 7 | DB | 15 | 51 | 40 | 27 | 7B | F3 | 74 | 3B | 98 |
| 181 | E | AB | 2A | A2 | 80 | 4E | F6 | FB | E8 | 76 | 2D |
| 182 | 1C | 4B | 54 | 59 | 1D | 9C | F1 | EB | CD | EC | 5A |
| 183 | 38 | 96 | A8 | B2 | 3A | 25 | FF | CB | 87 | C5 | B4 |
| 184 | 70 | 31 | 4D | 79 | 74 | 4A | E3 | 8B | 13 | 97 | 75 |
| 185 | E0 | 62 | 9A | F2 | E8 | 94 | DB | B | 26 | 33 | EA |
| 186 | DD | C4 | 29 | F9 | CD | 35 | AB | 16 | 4C | 66 | C9 |
| 187 | A7 | 95 | 52 | EF | 87 | 6A | 4B | 2C | 98 | CC | 8F |
| 188 | 53 | 37 | A4 | C3 | 13 | D4 | 96 | 58 | 2D | 85 | 3 |
| 189 | A6 | 6E | 55 | 9B | 26 | B5 | 31 | B0 | 5A | 17 | 6 |
| 190 | 51 | DC | AA | 2B | 4C | 77 | 62 | 7D | B4 | 2E | C |
| 191 | A2 | A5 | 49 | 56 | 98 | EE | C4 | FA | 75 | 5C | 18 |
| 192 | 59 | 57 | 92 | AC | 2D | C1 | 95 | E9 | EA | B8 | 30 |
| 193 | B2 | AE | 39 | 45 | 5A | 9F | 37 | CF | C9 | 6D | 60 |
| 194 | 79 | 41 | 72 | 8A | B4 | 23 | 6E | 83 | 8F | DA | C0 |
| 195 | F2 | 82 | E4 | 9 | 75 | 46 | DC | 1B | 3 | A9 | 9D |
| 196 | F9 | 19 | D5 | 12 | EA | 8C | A5 | 36 | 6 | 4F | 27 |
| 197 | EF | 32 | B7 | 24 | C9 | 5 | 57 | 6C | C | 9E | 4E |
| 198 | C3 | 64 | 73 | 48 | 8F | A | AE | D8 | 18 | 21 | 9C |
| 199 | 9B | C8 | E6 | 90 | 3 | 14 | 41 | AD | 30 | 42 | 25 |
| 200 | 2B | 8D | D1 | 3D | 6 | 28 | 82 | 47 | 60 | 84 | 4A |
| 201 | 56 | 7 | BF | 7A | C | 50 | 19 | 8E | C0 | 15 | 94 |
| 202 | AC | E | 63 | F4 | 18 | A0 | 32 | 02 | 9D | 2A | 35 |
| 203 | 45 | 1C | C6 | F5 | 30 | 5D | 64 | 04 | 27 | 54 | 6A |
| 204 | 8A | 38 | 91 | F7 | 60 | BA | C8 | 08 | 4E | A8 | D4 |
| 205 | 9 | 70 | 3F | F3 | C0 | 69 | 8D | 10 | 9C | 4D | B5 |
| 206 | 12 | E0 | 7E | FB | 9D | D2 | 7 | 20 | 25 | 9A | 77 |
| 207 | 24 | DD | FC | EB | 27 | B9 | E | 40 | 4A | 29 | EE |
| 208 | 48 | A7 | E5 | CB | 4E | 6F | 1C | 80 | 94 | 52 | C1 |
| 209 | 90 | 53 | D7 | 8B | 9C | DE | 38 | 1D | 35 | A4 | 9F |
| 210 | 3D | A6 | B3 | B | 25 | A1 | 70 | 3A | 6A | 55 | 23 |
| 211 | 7A | 51 | 7B | 16 | 4A | 5F | E0 | 74 | D4 | AA | 46 |
| 212 | F4 | A2 | F6 | 2C | 94 | BE | DD | E8 | B5 | 49 | 8C |
| 213 | F5 | 59 | F1 | 58 | 35 | 61 | A7 | CD | 77 | 92 | 5 |
| 214 | F7 | B2 | FF | B0 | 6A | C2 | 53 | 87 | EE | 39 | A |
| 215 | F3 | 79 | E3 | 7D | D4 | 99 | A6 | 13 | C1 | 72 | 14 |

| 216 | FB | F2 | DB | FA | B5 | 2F | 51 | 26 | 9F | E4 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 217 | EB | F9 | AB | E9 | 77 | 5E | A2 | 4C | 23 | D5 | 50 |
| 218 | CB | EF | 4B | CF | EE | BC | 59 | 98 | 46 | B7 | A0 |
| 219 | 8B | C3 | 96 | 83 | C1 | 65 | B2 | 2D | 8C | 73 | 5D |
| 220 | B | 9B | 31 | 1B | 9F | CA | 79 | 5A | 5 | E6 | BA |
| 221 | 16 | 2B | 62 | 36 | 23 | 89 | F2 | B4 | A | D1 | 69 |
| 222 | 2C | 56 | C4 | 6C | 46 | F | F9 | 75 | 14 | BF | D2 |
| 223 | 58 | AC | 95 | D8 | 8C | 1E | EF | EA | 28 | 63 | B9 |
| 224 | B0 | 45 | 37 | AD | 5 | 3C | C3 | C9 | 50 | C6 | 6F |
| 225 | 7D | 8A | 6E | 47 | A | 78 | 9B | 8F | A0 | 91 | DE |
| 226 | FA | 9 | DC | 8E | 14 | F0 | 2B | 3 | 5D | 3F | A1 |
| 227 | E9 | 12 | A5 | 02 | 28 | FD | 56 | 6 | BA | 7E | 5F |
| 228 | CF | 24 | 57 | 04 | 50 | E7 | AC | C | 69 | FC | BE |
| 229 | 83 | 48 | AE | 08 | A0 | D3 | 45 | 18 | D2 | E5 | 61 |
| 230 | 1B | 90 | 41 | 10 | 5D | BB | 8A | 30 | B9 | D7 | C2 |
| 231 | 36 | 3D | 82 | 20 | BA | 6B | 9 | 60 | 6F | B3 | 99 |
| 232 | 6C | 7A | 19 | 40 | 69 | D6 | 12 | C0 | DE | 7B | 2F |
| 233 | D8 | F4 | 32 | 80 | D2 | B1 | 24 | 9D | A1 | F6 | 5E |
| 234 | AD | F5 | 64 | 1D | B9 | 7F | 48 | 27 | 5F | F1 | BC |
| 235 | 47 | F7 | C8 | 3A | 6F | FE | 90 | 4E | BE | FF | 65 |
| 236 | 8E | F3 | 8D | 74 | DE | E1 | 3D | 9C | 61 | E3 | CA |
| 237 | 02 | FB | 7 | E8 | A1 | DF | 7A | 25 | C2 | DB | 89 |
| 238 | 04 | EB | E | CD | 5F | A3 | F4 | 4A | 99 | AB | F |
| 239 | 08 | CB | 1C | 87 | BE | 5B | F5 | 94 | 2F | 4B | 1E |
| 240 | 10 | 8B | 38 | 13 | 61 | B6 | F7 | 35 | 5E | 96 | 3C |
| 241 | 20 | B | 70 | 26 | C2 | 71 | F3 | 6A | BC | 31 | 78 |
| 242 | 40 | 16 | E0 | 4C | 99 | E2 | FB | D4 | 65 | 62 | F0 |
| 243 | 80 | 2C | DD | 98 | 2F | D9 | EB | B5 | CA | C4 | FD |
| 244 | 1D | 58 | A7 | 2D | 5E | AF | CB | 77 | 89 | 95 | E7 |
| 245 | 3A | B0 | 53 | 5A | BC | 43 | 8B | EE | F | 37 | D3 |
| 246 | 74 | 7D | A6 | B4 | 65 | 86 | B | C1 | 1E | 6E | BB |
| 247 | E8 | FA | 51 | 75 | CA | 11 | 16 | 9F | 3C | DC | 6B |
| 248 | CD | E9 | A2 | EA | 89 | 22 | 2C | 23 | 78 | A5 | D6 |
| 249 | 87 | CF | 59 | C9 | F | 44 | 58 | 46 | F0 | 57 | B1 |
| 250 | 13 | 83 | B2 | 8F | 1E | 88 | B0 | 8C | FD | AE | 7F |
| 251 | 26 | 1B | 79 | 3 | 3C | D | 7D | 5 | E7 | 41 | FE |
| 252 | 4C | 36 | F2 | 6 | 78 | 1A | FA | A | D3 | 82 | E1 |
| 253 | 98 | 6C | F9 | C | F0 | 34 | E9 | 14 | BB | 19 | DF |
| 254 | 2D | D8 | EF | 18 | FD | 68 | CF | 28 | 6B | 32 | A3 |

……..→Continue→……..

| Elem. GF (255) Of $\alpha^i$ | The ROMs Tables "HEX Form" Multiplying any elements from GF(255) with the relevant coefficients of the generator polynomial g(X) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $g_{11} = \alpha^{59}$ | $g_{12} = \alpha^{176}$ | $g_{13} = \alpha^{99}$ | $g_{14} = \alpha^{203}$ | $g_{15} = \alpha^{137}$ | $g_{16} = \alpha^{43}$ | $g_{17} = \alpha^{104}$ | $g_{18} = \alpha^{137}$ | $g_{19} = \alpha^{0}$ | $g_{20} = \alpha^{44}$ | $g_{21} = \alpha^{149}$ |
| - | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0 | D2 | E3 | 86 | E0 | 9E | 77 | D | 9E | 1 | EE | A4 |
| 1 | B9 | DB | 11 | DD | 21 | EE | 1A | 21 | 02 | C1 | 55 |
| 2 | 6F | AB | 22 | A7 | 42 | C1 | 34 | 42 | 04 | 9F | AA |
| 3 | DE | 4B | 44 | 53 | 84 | 9F | 68 | 84 | 08 | 23 | 49 |
| 4 | A1 | 96 | 88 | A6 | 15 | 23 | D0 | 15 | 10 | 46 | 92 |
| 5 | 5F | 31 | D | 51 | 2A | 46 | BD | 2A | 20 | 8C | 39 |
| 6 | BE | 62 | 1A | A2 | 54 | 8C | 67 | 54 | 40 | 5 | 72 |
| 7 | 61 | C4 | 34 | 59 | A8 | 5 | CE | A8 | 80 | A | E4 |
| 8 | C2 | 95 | 68 | B2 | 4D | A | 81 | 4D | 1D | 14 | D5 |
| 9 | 99 | 37 | D0 | 79 | 9A | 14 | 1F | 9A | 3A | 28 | B7 |
| 10 | 2F | 6E | BD | F2 | 29 | 28 | 3E | 29 | 74 | 50 | 73 |
| 11 | 5E | DC | 67 | F9 | 52 | 50 | 7C | 52 | E8 | A0 | E6 |
| 12 | BC | A5 | CE | EF | A4 | A0 | F8 | A4 | CD | 5D | D1 |
| 13 | 65 | 57 | 81 | C3 | 55 | 5D | ED | 55 | 87 | BA | BF |
| 14 | CA | AE | 1F | 9B | AA | BA | C7 | AA | 13 | 69 | 63 |
| 15 | 89 | 41 | 3E | 2B | 49 | 69 | 93 | 49 | 26 | D2 | C6 |
| 16 | F | 82 | 7C | 56 | 92 | D2 | 3B | 92 | 4C | B9 | 91 |
| 17 | 1E | 19 | F8 | AC | 39 | B9 | 76 | 39 | 98 | 6F | 3F |
| 18 | 3C | 32 | ED | 45 | 72 | 6F | EC | 72 | 2D | DE | 7E |
| 19 | 78 | 64 | C7 | 8A | E4 | DE | C5 | E4 | 5A | A1 | FC |
| 20 | F0 | C8 | 93 | 9 | D5 | A1 | 97 | D5 | B4 | 5F | E5 |
| 21 | FD | 8D | 3B | 12 | B7 | 5F | 33 | B7 | 75 | BE | D7 |
| 22 | E7 | 7 | 76 | 24 | 73 | BE | 66 | 73 | EA | 61 | B3 |
| 23 | D3 | E | EC | 48 | E6 | 61 | CC | E6 | C9 | C2 | 7B |
| 24 | BB | 1C | C5 | 90 | D1 | C2 | 85 | D1 | 8F | 99 | F6 |
| 25 | 6B | 38 | 97 | 3D | BF | 99 | 17 | BF | 3 | 2F | F1 |
| 26 | D6 | 70 | 33 | 7A | 63 | 2F | 2E | 63 | 6 | 5E | FF |
| 27 | B1 | E0 | 66 | F4 | C6 | 5E | 5C | C6 | C | BC | E3 |
| 28 | 7F | DD | CC | F5 | 91 | BC | B8 | 91 | 18 | 65 | DB |
| 29 | FE | A7 | 85 | F7 | 3F | 65 | 6D | 3F | 30 | CA | AB |
| 30 | E1 | 53 | 17 | F3 | 7E | CA | DA | 7E | 60 | 89 | 4B |
| 31 | DF | A6 | 2E | FB | FC | 89 | A9 | FC | C0 | F | 96 |
| 32 | A3 | 51 | 5C | EB | E5 | F | 4F | E5 | 9D | 1E | 31 |
| 33 | 5B | A2 | B8 | CB | D7 | 1E | 9E | D7 | 27 | 3C | 62 |
| 34 | B6 | 59 | 6D | 8B | B3 | 3C | 21 | B3 | 4E | 78 | C4 |
| 35 | 71 | B2 | DA | B | 7B | 78 | 42 | 7B | 9C | F0 | 95 |
| 36 | E2 | 79 | A9 | 16 | F6 | F0 | 84 | F6 | 25 | FD | 37 |
| 37 | D9 | F2 | 4F | 2C | F1 | FD | 15 | F1 | 4A | E7 | 6E |

| 38 | AF | F9 | 9E | 58 | FF | E7 | 2A | FF | 94 | D3 | DC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 39 | 43 | EF | 21 | B0 | E3 | D3 | 54 | E3 | 35 | BB | A5 |
| 40 | 86 | C3 | 42 | 7D | DB | BB | A8 | DB | 6A | 6B | 57 |
| 41 | 11 | 9B | 84 | FA | AB | 6B | 4D | AB | D4 | D6 | AE |
| 42 | 22 | 2B | 15 | E9 | 4B | D6 | 9A | 4B | B5 | B1 | 41 |
| 43 | 44 | 56 | 2A | CF | 96 | B1 | 29 | 96 | 77 | 7F | 82 |
| 44 | 88 | AC | 54 | 83 | 31 | 7F | 52 | 31 | EE | FE | 19 |
| 45 | D | 45 | A8 | 1B | 62 | FE | A4 | 62 | C1 | E1 | 32 |
| 46 | 1A | 8A | 4D | 36 | C4 | E1 | 55 | C4 | 9F | DF | 64 |
| 47 | 34 | 9 | 9A | 6C | 95 | DF | AA | 95 | 23 | A3 | C8 |
| 48 | 68 | 12 | 29 | D8 | 37 | A3 | 49 | 37 | 46 | 5B | 8D |
| 49 | D0 | 24 | 52 | AD | 6E | 5B | 92 | 6E | 8C | B6 | 7 |
| 50 | BD | 48 | A4 | 47 | DC | B6 | 39 | DC | 5 | 71 | E |
| 51 | 67 | 90 | 55 | 8E | A5 | 71 | 72 | A5 | A | E2 | 1C |
| 52 | CE | 3D | AA | 02 | 57 | E2 | E4 | 57 | 14 | D9 | 38 |
| 53 | 81 | 7A | 49 | 04 | AE | D9 | D5 | AE | 28 | AF | 70 |
| 54 | 1F | F4 | 92 | 08 | 41 | AF | B7 | 41 | 50 | 43 | E0 |
| 55 | 3E | F5 | 39 | 10 | 82 | 43 | 73 | 82 | A0 | 86 | DD |
| 56 | 7C | F7 | 72 | 20 | 19 | 86 | E6 | 19 | 5D | 11 | A7 |
| 57 | F8 | F3 | E4 | 40 | 32 | 11 | D1 | 32 | BA | 22 | 53 |
| 58 | ED | FB | D5 | 80 | 64 | 22 | BF | 64 | 69 | 44 | A6 |
| 59 | C7 | EB | B7 | 1D | C8 | 44 | 63 | C8 | D2 | 88 | 51 |
| 60 | 93 | CB | 73 | 3A | 8D | 88 | C6 | 8D | B9 | D | A2 |
| 61 | 3B | 8B | E6 | 74 | 7 | D | 91 | 7 | 6F | 1A | 59 |
| 62 | 76 | B | D1 | E8 | E | 1A | 3F | E | DE | 34 | B2 |
| 63 | EC | 16 | BF | CD | 1C | 34 | 7E | 1C | A1 | 68 | 79 |
| 64 | C5 | 2C | 63 | 87 | 38 | 68 | FC | 38 | 5F | D0 | F2 |
| 65 | 97 | 58 | C6 | 13 | 70 | D0 | E5 | 70 | BE | BD | F9 |
| 66 | 33 | B0 | 91 | 26 | E0 | BD | D7 | E0 | 61 | 67 | EF |
| 67 | 66 | 7D | 3F | 4C | DD | 67 | B3 | DD | C2 | CE | C3 |
| 68 | CC | FA | 7E | 98 | A7 | CE | 7B | A7 | 99 | 81 | 9B |
| 69 | 85 | E9 | FC | 2D | 53 | 81 | F6 | 53 | 2F | 1F | 2B |
| 70 | 17 | CF | E5 | 5A | A6 | 1F | F1 | A6 | 5E | 3E | 56 |
| 71 | 2E | 83 | D7 | B4 | 51 | 3E | FF | 51 | BC | 7C | AC |
| 72 | 5C | 1B | B3 | 75 | A2 | 7C | E3 | A2 | 65 | F8 | 45 |
| 73 | B8 | 36 | 7B | EA | 59 | F8 | DB | 59 | CA | ED | 8A |
| 74 | 6D | 6C | F6 | C9 | B2 | ED | AB | B2 | 89 | C7 | 9 |
| 75 | DA | D8 | F1 | 8F | 79 | C7 | 4B | 79 | F | 93 | 12 |
| 76 | A9 | AD | FF | 3 | F2 | 93 | 96 | F2 | 1E | 3B | 24 |
| 77 | 4F | 47 | E3 | 6 | F9 | 3B | 31 | F9 | 3C | 76 | 48 |
| 78 | 9E | 8E | DB | C | EF | 76 | 62 | EF | 78 | EC | 90 |
| 79 | 21 | 02 | AB | 18 | C3 | EC | C4 | C3 | F0 | C5 | 3D |
| 80 | 42 | 04 | 4B | 30 | 9B | C5 | 95 | 9B | FD | 97 | 7A |
| 81 | 84 | 08 | 96 | 60 | 2B | 97 | 37 | 2B | E7 | 33 | F4 |
| 82 | 15 | 10 | 31 | C0 | 56 | 33 | 6E | 56 | D3 | 66 | F5 |

| 83 | 2A | 20 | 62 | 9D | AC | 66 | DC | AC | BB | CC | F7 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 84 | 54 | 40 | C4 | 27 | 45 | CC | A5 | 45 | 6B | 85 | F3 |
| 85 | A8 | 80 | 95 | 4E | 8A | 85 | 57 | 8A | D6 | 17 | FB |
| 86 | 4D | 1D | 37 | 9C | 9 | 17 | AE | 9 | B1 | 2E | EB |
| 87 | 9A | 3A | 6E | 25 | 12 | 2E | 41 | 12 | 7F | 5C | CB |
| 88 | 29 | 74 | DC | 4A | 24 | 5C | 82 | 24 | FE | B8 | 8B |
| 89 | 52 | E8 | A5 | 94 | 48 | B8 | 19 | 48 | E1 | 6D | B |
| 90 | A4 | CD | 57 | 35 | 90 | 6D | 32 | 90 | DF | DA | 16 |
| 91 | 55 | 87 | AE | 6A | 3D | DA | 64 | 3D | A3 | A9 | 2C |
| 92 | AA | 13 | 41 | D4 | 7A | A9 | C8 | 7A | 5B | 4F | 58 |
| 93 | 49 | 26 | 82 | B5 | F4 | 4F | 8D | F4 | B6 | 9E | B0 |
| 94 | 92 | 4C | 19 | 77 | F5 | 9E | 7 | F5 | 71 | 21 | 7D |
| 95 | 39 | 98 | 32 | EE | F7 | 21 | E | F7 | E2 | 42 | FA |
| 96 | 72 | 2D | 64 | C1 | F3 | 42 | 1C | F3 | D9 | 84 | E9 |
| 97 | E4 | 5A | C8 | 9F | FB | 84 | 38 | FB | AF | 15 | CF |
| 98 | D5 | B4 | 8D | 23 | EB | 15 | 70 | EB | 43 | 2A | 83 |
| 99 | B7 | 75 | 7 | 46 | CB | 2A | E0 | CB | 86 | 54 | 1B |
| 100 | 73 | EA | E | 8C | 8B | 54 | DD | 8B | 11 | A8 | 36 |
| 101 | E6 | C9 | 1C | 5 | B | A8 | A7 | B | 22 | 4D | 6C |
| 102 | D1 | 8F | 38 | A | 16 | 4D | 53 | 16 | 44 | 9A | D8 |
| 103 | BF | 3 | 70 | 14 | 2C | 9A | A6 | 2C | 88 | 29 | AD |
| 104 | 63 | 6 | E0 | 28 | 58 | 29 | 51 | 58 | D | 52 | 47 |
| 105 | C6 | C | DD | 50 | B0 | 52 | A2 | B0 | 1A | A4 | 8E |
| 106 | 91 | 18 | A7 | A0 | 7D | A4 | 59 | 7D | 34 | 55 | 02 |
| 107 | 3F | 30 | 53 | 5D | FA | 55 | B2 | FA | 68 | AA | 04 |
| 108 | 7E | 60 | A6 | BA | E9 | AA | 79 | E9 | D0 | 49 | 08 |
| 109 | FC | C0 | 51 | 69 | CF | 49 | F2 | CF | BD | 92 | 10 |
| 110 | E5 | 9D | A2 | D2 | 83 | 92 | F9 | 83 | 67 | 39 | 20 |
| 111 | D7 | 27 | 59 | B9 | 1B | 39 | EF | 1B | CE | 72 | 40 |
| 112 | B3 | 4E | B2 | 6F | 36 | 72 | C3 | 36 | 81 | E4 | 80 |
| 113 | 7B | 9C | 79 | DE | 6C | E4 | 9B | 6C | 1F | D5 | 1D |
| 114 | F6 | 25 | F2 | A1 | D8 | D5 | 2B | D8 | 3E | B7 | 3A |
| 115 | F1 | 4A | F9 | 5F | AD | B7 | 56 | AD | 7C | 73 | 74 |
| 116 | FF | 94 | EF | BE | 47 | 73 | AC | 47 | F8 | E6 | E8 |
| 117 | E3 | 35 | C3 | 61 | 8E | E6 | 45 | 8E | ED | D1 | CD |
| 118 | DB | 6A | 9B | C2 | 02 | D1 | 8A | 02 | C7 | BF | 87 |
| 119 | AB | D4 | 2B | 99 | 04 | BF | 9 | 04 | 93 | 63 | 13 |
| 120 | 4B | B5 | 56 | 2F | 08 | 63 | 12 | 08 | 3B | C6 | 26 |
| 121 | 96 | 77 | AC | 5E | 10 | C6 | 24 | 10 | 76 | 91 | 4C |
| 122 | 31 | EE | 45 | BC | 20 | 91 | 48 | 20 | EC | 3F | 98 |
| 123 | 62 | C1 | 8A | 65 | 40 | 3F | 90 | 40 | C5 | 7E | 2D |
| 124 | C4 | 9F | 9 | CA | 80 | 7E | 3D | 80 | 97 | FC | 5A |
| 125 | 95 | 23 | 12 | 89 | 1D | FC | 7A | 1D | 33 | E5 | B4 |
| 126 | 37 | 46 | 24 | F | 3A | E5 | F4 | 3A | 66 | D7 | 75 |
| 127 | 6E | 8C | 48 | 1E | 74 | D7 | F5 | 74 | CC | B3 | EA |

| 128 | DC | 5 | 90 | 3C | E8 | B3 | F7 | E8 | 85 | 7B | C9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 129 | A5 | A | 3D | 78 | CD | 7B | F3 | CD | 17 | F6 | 8F |
| 130 | 57 | 14 | 7A | F0 | 87 | F6 | FB | 87 | 2E | F1 | 3 |
| 131 | AE | 28 | F4 | FD | 13 | F1 | EB | 13 | 5C | FF | 6 |
| 132 | 41 | 50 | F5 | E7 | 26 | FF | CB | 26 | B8 | E3 | C |
| 133 | 82 | A0 | F7 | D3 | 4C | E3 | 8B | 4C | 6D | DB | 18 |
| 134 | 19 | 5D | F3 | BB | 98 | DB | B | 98 | DA | AB | 30 |
| 135 | 32 | BA | FB | 6B | 2D | AB | 16 | 2D | A9 | 4B | 60 |
| 136 | 64 | 69 | EB | D6 | 5A | 4B | 2C | 5A | 4F | 96 | C0 |
| 137 | C8 | D2 | CB | B1 | B4 | 96 | 58 | B4 | 9E | 31 | 9D |
| 138 | 8D | B9 | 8B | 7F | 75 | 31 | B0 | 75 | 21 | 62 | 27 |
| 139 | 7 | 6F | B | FE | EA | 62 | 7D | EA | 42 | C4 | 4E |
| 140 | E | DE | 16 | E1 | C9 | C4 | FA | C9 | 84 | 95 | 9C |
| 141 | 1C | A1 | 2C | DF | 8F | 95 | E9 | 8F | 15 | 37 | 25 |
| 142 | 38 | 5F | 58 | A3 | 3 | 37 | CF | 3 | 2A | 6E | 4A |
| 143 | 70 | BE | B0 | 5B | 6 | 6E | 83 | 6 | 54 | DC | 94 |
| 144 | E0 | 61 | 7D | B6 | C | DC | 1B | C | A8 | A5 | 35 |
| 145 | DD | C2 | FA | 71 | 18 | A5 | 36 | 18 | 4D | 57 | 6A |
| 146 | A7 | 99 | E9 | E2 | 30 | 57 | 6C | 30 | 9A | AE | D4 |
| 147 | 53 | 2F | CF | D9 | 60 | AE | D8 | 60 | 29 | 41 | B5 |
| 148 | A6 | 5E | 83 | AF | C0 | 41 | AD | C0 | 52 | 82 | 77 |
| 149 | 51 | BC | 1B | 43 | 9D | 82 | 47 | 9D | A4 | 19 | EE |
| 150 | A2 | 65 | 36 | 86 | 27 | 19 | 8E | 27 | 55 | 32 | C1 |
| 151 | 59 | CA | 6C | 11 | 4E | 32 | 02 | 4E | AA | 64 | 9F |
| 152 | B2 | 89 | D8 | 22 | 9C | 64 | 04 | 9C | 49 | C8 | 23 |
| 153 | 79 | F | AD | 44 | 25 | C8 | 08 | 25 | 92 | 8D | 46 |
| 154 | F2 | 1E | 47 | 88 | 4A | 8D | 10 | 4A | 39 | 7 | 8C |
| 155 | F9 | 3C | 8E | D | 94 | 7 | 20 | 94 | 72 | E | 5 |
| 156 | EF | 78 | 02 | 1A | 35 | E | 40 | 35 | E4 | 1C | A |
| 157 | C3 | F0 | 04 | 34 | 6A | 1C | 80 | 6A | D5 | 38 | 14 |
| 158 | 9B | FD | 08 | 68 | D4 | 38 | 1D | D4 | B7 | 70 | 28 |
| 159 | 2B | E7 | 10 | D0 | B5 | 70 | 3A | B5 | 73 | E0 | 50 |
| 160 | 56 | D3 | 20 | BD | 77 | E0 | 74 | 77 | E6 | DD | A0 |
| 161 | AC | BB | 40 | 67 | EE | DD | E8 | EE | D1 | A7 | 5D |
| 162 | 45 | 6B | 80 | CE | C1 | A7 | CD | C1 | BF | 53 | BA |
| 163 | 8A | D6 | 1D | 81 | 9F | 53 | 87 | 9F | 63 | A6 | 69 |
| 164 | 9 | B1 | 3A | 1F | 23 | A6 | 13 | 23 | C6 | 51 | D2 |
| 165 | 12 | 7F | 74 | 3E | 46 | 51 | 26 | 46 | 91 | A2 | B9 |
| 166 | 24 | FE | E8 | 7C | 8C | A2 | 4C | 8C | 3F | 59 | 6F |
| 167 | 48 | E1 | CD | F8 | 5 | 59 | 98 | 5 | 7E | B2 | DE |
| 168 | 90 | DF | 87 | ED | A | B2 | 2D | A | FC | 79 | A1 |
| 169 | 3D | A3 | 13 | C7 | 14 | 79 | 5A | 14 | E5 | F2 | 5F |
| 170 | 7A | 5B | 26 | 93 | 28 | F2 | B4 | 28 | D7 | F9 | BE |
| 171 | F4 | B6 | 4C | 3B | 50 | F9 | 75 | 50 | B3 | EF | 61 |
| 172 | F5 | 71 | 98 | 76 | A0 | EF | EA | A0 | 7B | C3 | C2 |

| 173 | F7 | E2 | 2D | EC | 5D | C3 | C9 | 5D | F6 | 9B | 99 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 174 | F3 | D9 | 5A | C5 | BA | 9B | 8F | BA | F1 | 2B | 2F |
| 175 | FB | AF | B4 | 97 | 69 | 2B | 3 | 69 | FF | 56 | 5E |
| 176 | EB | 43 | 75 | 33 | D2 | 56 | 6 | D2 | E3 | AC | BC |
| 177 | CB | 86 | EA | 66 | B9 | AC | C | B9 | DB | 45 | 65 |
| 178 | 8B | 11 | C9 | CC | 6F | 45 | 18 | 6F | AB | 8A | CA |
| 179 | B | 22 | 8F | 85 | DE | 8A | 30 | DE | 4B | 9 | 89 |
| 180 | 16 | 44 | 3 | 17 | A1 | 9 | 60 | A1 | 96 | 12 | F |
| 181 | 2C | 88 | 6 | 2E | 5F | 12 | C0 | 5F | 31 | 24 | 1E |
| 182 | 58 | D | C | 5C | BE | 24 | 9D | BE | 62 | 48 | 3C |
| 183 | B0 | 1A | 18 | B8 | 61 | 48 | 27 | 61 | C4 | 90 | 78 |
| 184 | 7D | 34 | 30 | 6D | C2 | 90 | 4E | C2 | 95 | 3D | F0 |
| 185 | FA | 68 | 60 | DA | 99 | 3D | 9C | 99 | 37 | 7A | FD |
| 186 | E9 | D0 | C0 | A9 | 2F | 7A | 25 | 2F | 6E | F4 | E7 |
| 187 | CF | BD | 9D | 4F | 5E | F4 | 4A | 5E | DC | F5 | D3 |
| 188 | 83 | 67 | 27 | 9E | BC | F5 | 94 | BC | A5 | F7 | BB |
| 189 | 1B | CE | 4E | 21 | 65 | F7 | 35 | 65 | 57 | F3 | 6B |
| 190 | 36 | 81 | 9C | 42 | CA | F3 | 6A | CA | AE | FB | D6 |
| 191 | 6C | 1F | 25 | 84 | 89 | FB | D4 | 89 | 41 | EB | B1 |
| 192 | D8 | 3E | 4A | 15 | F | EB | B5 | F | 82 | CB | 7F |
| 193 | AD | 7C | 94 | 2A | 1E | CB | 77 | 1E | 19 | 8B | FE |
| 194 | 47 | F8 | 35 | 54 | 3C | 8B | EE | 3C | 32 | B | E1 |
| 195 | 8E | ED | 6A | A8 | 78 | B | C1 | 78 | 64 | 16 | DF |
| 196 | 02 | C7 | D4 | 4D | F0 | 16 | 9F | F0 | C8 | 2C | A3 |
| 197 | 04 | 93 | B5 | 9A | FD | 2C | 23 | FD | 8D | 58 | 5B |
| 198 | 08 | 3B | 77 | 29 | E7 | 58 | 46 | E7 | 7 | B0 | B6 |
| 199 | 10 | 76 | EE | 52 | D3 | B0 | 8C | D3 | E | 7D | 71 |
| 200 | 20 | EC | C1 | A4 | BB | 7D | 5 | BB | 1C | FA | E2 |
| 201 | 40 | C5 | 9F | 55 | 6B | FA | A | 6B | 38 | E9 | D9 |
| 202 | 80 | 97 | 23 | AA | D6 | E9 | 14 | D6 | 70 | CF | AF |
| 203 | 1D | 33 | 46 | 49 | B1 | CF | 28 | B1 | E0 | 83 | 43 |
| 204 | 3A | 66 | 8C | 92 | 7F | 83 | 50 | 7F | DD | 1B | 86 |
| 205 | 74 | CC | 5 | 39 | FE | 1B | A0 | FE | A7 | 36 | 11 |
| 206 | E8 | 85 | A | 72 | E1 | 36 | 5D | E1 | 53 | 6C | 22 |
| 207 | CD | 17 | 14 | E4 | DF | 6C | BA | DF | A6 | D8 | 44 |
| 208 | 87 | 2E | 28 | D5 | A3 | D8 | 69 | A3 | 51 | AD | 88 |
| 209 | 13 | 5C | 50 | B7 | 5B | AD | D2 | 5B | A2 | 47 | D |
| 210 | 26 | B8 | A0 | 73 | B6 | 47 | B9 | B6 | 59 | 8E | 1A |
| 211 | 4C | 6D | 5D | E6 | 71 | 8E | 6F | 71 | B2 | 02 | 34 |
| 212 | 98 | DA | BA | D1 | E2 | 02 | DE | E2 | 79 | 04 | 68 |
| 213 | 2D | A9 | 69 | BF | D9 | 04 | A1 | D9 | F2 | 08 | D0 |
| 214 | 5A | 4F | D2 | 63 | AF | 08 | 5F | AF | F9 | 10 | BD |
| 215 | B4 | 9E | B9 | C6 | 43 | 10 | BE | 43 | EF | 20 | 67 |
| 216 | 75 | 21 | 6F | 91 | 86 | 20 | 61 | 86 | C3 | 40 | CE |
| 217 | EA | 42 | DE | 3F | 11 | 40 | C2 | 11 | 9B | 80 | 81 |

| 218 | C9 | 84 | A1 | 7E | 22 | 80 | 99 | 22 | 2B | 1D | 1F |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 219 | 8F | 15 | 5F | FC | 44 | 1D | 2F | 44 | 56 | 3A | 3E |
| 220 | 3 | 2A | BE | E5 | 88 | 3A | 5E | 88 | AC | 74 | 7C |
| 221 | 6 | 54 | 61 | D7 | D | 74 | BC | D | 45 | E8 | F8 |
| 222 | C | A8 | C2 | B3 | 1A | E8 | 65 | 1A | 8A | CD | ED |
| 223 | 18 | 4D | 99 | 7B | 34 | CD | CA | 34 | 9 | 87 | C7 |
| 224 | 30 | 9A | 2F | F6 | 68 | 87 | 89 | 68 | 12 | 13 | 93 |
| 225 | 60 | 29 | 5E | F1 | D0 | 13 | F | D0 | 24 | 26 | 3B |
| 226 | C0 | 52 | BC | FF | BD | 26 | 1E | BD | 48 | 4C | 76 |
| 227 | 9D | A4 | 65 | E3 | 67 | 4C | 3C | 67 | 90 | 98 | EC |
| 228 | 27 | 55 | CA | DB | CE | 98 | 78 | CE | 3D | 2D | C5 |
| 229 | 4E | AA | 89 | AB | 81 | 2D | F0 | 81 | 7A | 5A | 97 |
| 230 | 9C | 49 | F | 4B | 1F | 5A | FD | 1F | F4 | B4 | 33 |
| 231 | 25 | 92 | 1E | 96 | 3E | B4 | E7 | 3E | F5 | 75 | 66 |
| 232 | 4A | 39 | 3C | 31 | 7C | 75 | D3 | 7C | F7 | EA | CC |
| 233 | 94 | 72 | 78 | 62 | F8 | EA | BB | F8 | F3 | C9 | 85 |
| 234 | 35 | E4 | F0 | C4 | ED | C9 | 6B | ED | FB | 8F | 17 |
| 235 | 6A | D5 | FD | 95 | C7 | 8F | D6 | C7 | EB | 3 | 2E |
| 236 | D4 | B7 | E7 | 37 | 93 | 3 | B1 | 93 | CB | 6 | 5C |
| 237 | B5 | 73 | D3 | 6E | 3B | 6 | 7F | 3B | 8B | C | B8 |
| 238 | 77 | E6 | BB | DC | 76 | C | FE | 76 | B | 18 | 6D |
| 239 | EE | D1 | 6B | A5 | EC | 18 | E1 | EC | 16 | 30 | DA |
| 240 | C1 | BF | D6 | 57 | C5 | 30 | DF | C5 | 2C | 60 | A9 |
| 241 | 9F | 63 | B1 | AE | 97 | 60 | A3 | 97 | 58 | C0 | 4F |
| 242 | 23 | C6 | 7F | 41 | 33 | C0 | 5B | 33 | B0 | 9D | 9E |
| 243 | 46 | 91 | FE | 82 | 66 | 9D | B6 | 66 | 7D | 27 | 21 |
| 244 | 8C | 3F | E1 | 19 | CC | 27 | 71 | CC | FA | 4E | 42 |
| 245 | 5 | 7E | DF | 32 | 85 | 4E | E2 | 85 | E9 | 9C | 84 |
| 246 | A | FC | A3 | 64 | 17 | 9C | D9 | 17 | CF | 25 | 15 |
| 247 | 14 | E5 | 5B | C8 | 2E | 25 | AF | 2E | 83 | 4A | 2A |
| 248 | 28 | D7 | B6 | 8D | 5C | 4A | 43 | 5C | 1B | 94 | 54 |
| 249 | 50 | B3 | 71 | 7 | B8 | 94 | 86 | B8 | 36 | 35 | A8 |
| 250 | A0 | 7B | E2 | E | 6D | 35 | 11 | 6D | 6C | 6A | 4D |
| 251 | 5D | F6 | D9 | 1C | DA | 6A | 22 | DA | D8 | D4 | 9A |
| 252 | BA | F1 | AF | 38 | A9 | D4 | 44 | A9 | AD | B5 | 29 |
| 253 | 69 | FF | 43 | 70 | 4F | B5 | 88 | 4F | 47 | 77 | 52 |
| 254 | D2 | E3 | 86 | E0 | 9E | 77 | D | 9E | 8E | EE | A4 |

……..→Continue→……..……..

| Elem. GF (255) Of $\alpha^i$ | The ROMs Tables "HEX Form" Multiplying any elements from GF(255) with the relevant coefficients of the generator polynomial g(X) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $g_{22} = \alpha^{148}$ | $g_{23} = \alpha^{218}$ | $g_{24} = \alpha^{75}$ | $g_{25} = \alpha^{11}$ | $g_{26} = \alpha^{173}$ | $g_{27} = \alpha^{254}$ | $g_{28} = \alpha^{194}$ | $g_{29} = \alpha^{109}$ | $g_{30} = \alpha^{8}$ | $g_{31} = \alpha^{11}$ |
| - | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0 | 52 | 2B | F | E8 | F6 | 8E | 32 | BD | 1D | E8 |
| 1 | A4 | 56 | 1E | CD | F1 | 02 | 64 | 67 | 3A | CD |
| 2 | 55 | AC | 3C | 87 | FF | 04 | C8 | CE | 74 | 87 |
| 3 | AA | 45 | 78 | 13 | E3 | 08 | 8D | 81 | E8 | 13 |
| 4 | 49 | 8A | F0 | 26 | DB | 10 | 7 | 1F | CD | 26 |
| 5 | 92 | 9 | FD | 4C | AB | 20 | E | 3E | 87 | 4C |
| 6 | 39 | 12 | E7 | 98 | 4B | 40 | 1C | 7C | 13 | 98 |
| 7 | 72 | 24 | D3 | 2D | 96 | 80 | 38 | F8 | 26 | 2D |
| 8 | E4 | 48 | BB | 5A | 31 | 1D | 70 | ED | 4C | 5A |
| 9 | D5 | 90 | 6B | B4 | 62 | 3A | E0 | C7 | 98 | B4 |
| 10 | B7 | 3D | D6 | 75 | C4 | 74 | DD | 93 | 2D | 75 |
| 11 | 73 | 7A | B1 | EA | 95 | E8 | A7 | 3B | 5A | EA |
| 12 | E6 | F4 | 7F | C9 | 37 | CD | 53 | 76 | B4 | C9 |
| 13 | D1 | F5 | FE | 8F | 6E | 87 | A6 | EC | 75 | 8F |
| 14 | BF | F7 | E1 | 3 | DC | 13 | 51 | C5 | EA | 3 |
| 15 | 63 | F3 | DF | 6 | A5 | 26 | A2 | 97 | C9 | 6 |
| 16 | C6 | FB | A3 | C | 57 | 4C | 59 | 33 | 8F | C |
| 17 | 91 | EB | 5B | 18 | AE | 98 | B2 | 66 | 3 | 18 |
| 18 | 3F | CB | B6 | 30 | 41 | 2D | 79 | CC | 6 | 30 |
| 19 | 7E | 8B | 71 | 60 | 82 | 5A | F2 | 85 | C | 60 |
| 20 | FC | B | E2 | C0 | 19 | B4 | F9 | 17 | 18 | C0 |
| 21 | E5 | 16 | D9 | 9D | 32 | 75 | EF | 2E | 30 | 9D |
| 22 | D7 | 2C | AF | 27 | 64 | EA | C3 | 5C | 60 | 27 |
| 23 | B3 | 58 | 43 | 4E | C8 | C9 | 9B | B8 | C0 | 4E |
| 24 | 7B | B0 | 86 | 9C | 8D | 8F | 2B | 6D | 9D | 9C |
| 25 | F6 | 7D | 11 | 25 | 7 | 3 | 56 | DA | 27 | 25 |
| 26 | F1 | FA | 22 | 4A | E | 6 | AC | A9 | 4E | 4A |
| 27 | FF | E9 | 44 | 94 | 1C | C | 45 | 4F | 9C | 94 |
| 28 | E3 | CF | 88 | 35 | 38 | 18 | 8A | 9E | 25 | 35 |
| 29 | DB | 83 | D | 6A | 70 | 30 | 9 | 21 | 4A | 6A |
| 30 | AB | 1B | 1A | D4 | E0 | 60 | 12 | 42 | 94 | D4 |
| 31 | 4B | 36 | 34 | B5 | DD | C0 | 24 | 84 | 35 | B5 |
| 32 | 96 | 6C | 68 | 77 | A7 | 9D | 48 | 15 | 6A | 77 |
| 33 | 31 | D8 | D0 | EE | 53 | 27 | 90 | 2A | D4 | EE |
| 34 | 62 | AD | BD | C1 | A6 | 4E | 3D | 54 | B5 | C1 |
| 35 | C4 | 47 | 67 | 9F | 51 | 9C | 7A | A8 | 77 | 9F |
| 36 | 95 | 8E | CE | 23 | A2 | 25 | F4 | 4D | EE | 23 |
| 37 | 37 | 02 | 81 | 46 | 59 | 4A | F5 | 9A | C1 | 46 |

| 38 | 6E | 04 | 1F | 8C | B2 | 94 | F7 | 29 | 9F | 8C |
|---|---|---|---|---|---|---|---|---|---|---|
| 39 | DC | 08 | 3E | 5 | 79 | 35 | F3 | 52 | 23 | 5 |
| 40 | A5 | 10 | 7C | A | F2 | 6A | FB | A4 | 46 | A |
| 41 | 57 | 20 | F8 | 14 | F9 | D4 | EB | 55 | 8C | 14 |
| 42 | AE | 40 | ED | 28 | EF | B5 | CB | AA | 5 | 28 |
| 43 | 41 | 80 | C7 | 50 | C3 | 77 | 8B | 49 | A | 50 |
| 44 | 82 | 1D | 93 | A0 | 9B | EE | B | 92 | 14 | A0 |
| 45 | 19 | 3A | 3B | 5D | 2B | C1 | 16 | 39 | 28 | 5D |
| 46 | 32 | 74 | 76 | BA | 56 | 9F | 2C | 72 | 50 | BA |
| 47 | 64 | E8 | EC | 69 | AC | 23 | 58 | E4 | A0 | 69 |
| 48 | C8 | CD | C5 | D2 | 45 | 46 | B0 | D5 | 5D | D2 |
| 49 | 8D | 87 | 97 | B9 | 8A | 8C | 7D | B7 | BA | B9 |
| 50 | 7 | 13 | 33 | 6F | 9 | 5 | FA | 73 | 69 | 6F |
| 51 | E | 26 | 66 | DE | 12 | A | E9 | E6 | D2 | DE |
| 52 | 1C | 4C | CC | A1 | 24 | 14 | CF | D1 | B9 | A1 |
| 53 | 38 | 98 | 85 | 5F | 48 | 28 | 83 | BF | 6F | 5F |
| 54 | 70 | 2D | 17 | BE | 90 | 50 | 1B | 63 | DE | BE |
| 55 | E0 | 5A | 2E | 61 | 3D | A0 | 36 | C6 | A1 | 61 |
| 56 | DD | B4 | 5C | C2 | 7A | 5D | 6C | 91 | 5F | C2 |
| 57 | A7 | 75 | B8 | 99 | F4 | BA | D8 | 3F | BE | 99 |
| 58 | 53 | EA | 6D | 2F | F5 | 69 | AD | 7E | 61 | 2F |
| 59 | A6 | C9 | DA | 5E | F7 | D2 | 47 | FC | C2 | 5E |
| 60 | 51 | 8F | A9 | BC | F3 | B9 | 8E | E5 | 99 | BC |
| 61 | A2 | 3 | 4F | 65 | FB | 6F | 02 | D7 | 2F | 65 |
| 62 | 59 | 6 | 9E | CA | EB | DE | 04 | B3 | 5E | CA |
| 63 | B2 | C | 21 | 89 | CB | A1 | 08 | 7B | BC | 89 |
| 64 | 79 | 18 | 42 | F | 8B | 5F | 10 | F6 | 65 | F |
| 65 | F2 | 30 | 84 | 1E | B | BE | 20 | F1 | CA | 1E |
| 66 | F9 | 60 | 15 | 3C | 16 | 61 | 40 | FF | 89 | 3C |
| 67 | EF | C0 | 2A | 78 | 2C | C2 | 80 | E3 | F | 78 |
| 68 | C3 | 9D | 54 | F0 | 58 | 99 | 1D | DB | 1E | F0 |
| 69 | 9B | 27 | A8 | FD | B0 | 2F | 3A | AB | 3C | FD |
| 70 | 2B | 4E | 4D | E7 | 7D | 5E | 74 | 4B | 78 | E7 |
| 71 | 56 | 9C | 9A | D3 | FA | BC | E8 | 96 | F0 | D3 |
| 72 | AC | 25 | 29 | BB | E9 | 65 | CD | 31 | FD | BB |
| 73 | 45 | 4A | 52 | 6B | CF | CA | 87 | 62 | E7 | 6B |
| 74 | 8A | 94 | A4 | D6 | 83 | 89 | 13 | C4 | D3 | D6 |
| 75 | 9 | 35 | 55 | B1 | 1B | F | 26 | 95 | BB | B1 |
| 76 | 12 | 6A | AA | 7F | 36 | 1E | 4C | 37 | 6B | 7F |
| 77 | 24 | D4 | 49 | FE | 6C | 3C | 98 | 6E | D6 | FE |
| 78 | 48 | B5 | 92 | E1 | D8 | 78 | 2D | DC | B1 | E1 |
| 79 | 90 | 77 | 39 | DF | AD | F0 | 5A | A5 | 7F | DF |
| 80 | 3D | EE | 72 | A3 | 47 | FD | B4 | 57 | FE | A3 |
| 81 | 7A | C1 | E4 | 5B | 8E | E7 | 75 | AE | E1 | 5B |
| 82 | F4 | 9F | D5 | B6 | 02 | D3 | EA | 41 | DF | B6 |

| 83 | F5 | 23 | B7 | 71 | 04 | BB | C9 | 82 | A3 | 71 |
|---|---|---|---|---|---|---|---|---|---|---|
| 84 | F7 | 46 | 73 | E2 | 08 | 6B | 8F | 19 | 5B | E2 |
| 85 | F3 | 8C | E6 | D9 | 10 | D6 | 3 | 32 | B6 | D9 |
| 86 | FB | 5 | D1 | AF | 20 | B1 | 6 | 64 | 71 | AF |
| 87 | EB | A | BF | 43 | 40 | 7F | C | C8 | E2 | 43 |
| 88 | CB | 14 | 63 | 86 | 80 | FE | 18 | 8D | D9 | 86 |
| 89 | 8B | 28 | C6 | 11 | 1D | E1 | 30 | 7 | AF | 11 |
| 90 | B | 50 | 91 | 22 | 3A | DF | 60 | E | 43 | 22 |
| 91 | 16 | A0 | 3F | 44 | 74 | A3 | C0 | 1C | 86 | 44 |
| 92 | 2C | 5D | 7E | 88 | E8 | 5B | 9D | 38 | 11 | 88 |
| 93 | 58 | BA | FC | D | CD | B6 | 27 | 70 | 22 | D |
| 94 | B0 | 69 | E5 | 1A | 87 | 71 | 4E | E0 | 44 | 1A |
| 95 | 7D | D2 | D7 | 34 | 13 | E2 | 9C | DD | 88 | 34 |
| 96 | FA | B9 | B3 | 68 | 26 | D9 | 25 | A7 | D | 68 |
| 97 | E9 | 6F | 7B | D0 | 4C | AF | 4A | 53 | 1A | D0 |
| 98 | CF | DE | F6 | BD | 98 | 43 | 94 | A6 | 34 | BD |
| 99 | 83 | A1 | F1 | 67 | 2D | 86 | 35 | 51 | 68 | 67 |
| 100 | 1B | 5F | FF | CE | 5A | 11 | 6A | A2 | D0 | CE |
| 101 | 36 | BE | E3 | 81 | B4 | 22 | D4 | 59 | BD | 81 |
| 102 | 6C | 61 | DB | 1F | 75 | 44 | B5 | B2 | 67 | 1F |
| 103 | D8 | C2 | AB | 3E | EA | 88 | 77 | 79 | CE | 3E |
| 104 | AD | 99 | 4B | 7C | C9 | D | EE | F2 | 81 | 7C |
| 105 | 47 | 2F | 96 | F8 | 8F | 1A | C1 | F9 | 1F | F8 |
| 106 | 8E | 5E | 31 | ED | 3 | 34 | 9F | EF | 3E | ED |
| 107 | 02 | BC | 62 | C7 | 6 | 68 | 23 | C3 | 7C | C7 |
| 108 | 04 | 65 | C4 | 93 | C | D0 | 46 | 9B | F8 | 93 |
| 109 | 08 | CA | 95 | 3B | 18 | BD | 8C | 2B | ED | 3B |
| 110 | 10 | 89 | 37 | 76 | 30 | 67 | 5 | 56 | C7 | 76 |
| 111 | 20 | F | 6E | EC | 60 | CE | A | AC | 93 | EC |
| 112 | 40 | 1E | DC | C5 | C0 | 81 | 14 | 45 | 3B | C5 |
| 113 | 80 | 3C | A5 | 97 | 9D | 1F | 28 | 8A | 76 | 97 |
| 114 | 1D | 78 | 57 | 33 | 27 | 3E | 50 | 9 | EC | 33 |
| 115 | 3A | F0 | AE | 66 | 4E | 7C | A0 | 12 | C5 | 66 |
| 116 | 74 | FD | 41 | CC | 9C | F8 | 5D | 24 | 97 | CC |
| 117 | E8 | E7 | 82 | 85 | 25 | ED | BA | 48 | 33 | 85 |
| 118 | CD | D3 | 19 | 17 | 4A | C7 | 69 | 90 | 66 | 17 |
| 119 | 87 | BB | 32 | 2E | 94 | 93 | D2 | 3D | CC | 2E |
| 120 | 13 | 6B | 64 | 5C | 35 | 3B | B9 | 7A | 85 | 5C |
| 121 | 26 | D6 | C8 | B8 | 6A | 76 | 6F | F4 | 17 | B8 |
| 122 | 4C | B1 | 8D | 6D | D4 | EC | DE | F5 | 2E | 6D |
| 123 | 98 | 7F | 7 | DA | B5 | C5 | A1 | F7 | 5C | DA |
| 124 | 2D | FE | E | A9 | 77 | 97 | 5F | F3 | B8 | A9 |
| 125 | 5A | E1 | 1C | 4F | EE | 33 | BE | FB | 6D | 4F |
| 126 | B4 | DF | 38 | 9E | C1 | 66 | 61 | EB | DA | 9E |
| 127 | 75 | A3 | 70 | 21 | 9F | CC | C2 | CB | A9 | 21 |

| 128 | EA | 5B | E0 | 42 | 23 | 85 | 99 | 8B | 4F | 42 |
|---|---|---|---|---|---|---|---|---|---|---|
| 129 | C9 | B6 | DD | 84 | 46 | 17 | 2F | B | 9E | 84 |
| 130 | 8F | 71 | A7 | 15 | 8C | 2E | 5E | 16 | 21 | 15 |
| 131 | 3 | E2 | 53 | 2A | 5 | 5C | BC | 2C | 42 | 2A |
| 132 | 6 | D9 | A6 | 54 | A | B8 | 65 | 58 | 84 | 54 |
| 133 | C | AF | 51 | A8 | 14 | 6D | CA | B0 | 15 | A8 |
| 134 | 18 | 43 | A2 | 4D | 28 | DA | 89 | 7D | 2A | 4D |
| 135 | 30 | 86 | 59 | 9A | 50 | A9 | F | FA | 54 | 9A |
| 136 | 60 | 11 | B2 | 29 | A0 | 4F | 1E | E9 | A8 | 29 |
| 137 | C0 | 22 | 79 | 52 | 5D | 9E | 3C | CF | 4D | 52 |
| 138 | 9D | 44 | F2 | A4 | BA | 21 | 78 | 83 | 9A | A4 |
| 139 | 27 | 88 | F9 | 55 | 69 | 42 | F0 | 1B | 29 | 55 |
| 140 | 4E | D | EF | AA | D2 | 84 | FD | 36 | 52 | AA |
| 141 | 9C | 1A | C3 | 49 | B9 | 15 | E7 | 6C | A4 | 49 |
| 142 | 25 | 34 | 9B | 92 | 6F | 2A | D3 | D8 | 55 | 92 |
| 143 | 4A | 68 | 2B | 39 | DE | 54 | BB | AD | AA | 39 |
| 144 | 94 | D0 | 56 | 72 | A1 | A8 | 6B | 47 | 49 | 72 |
| 145 | 35 | BD | AC | E4 | 5F | 4D | D6 | 8E | 92 | E4 |
| 146 | 6A | 67 | 45 | D5 | BE | 9A | B1 | 02 | 39 | D5 |
| 147 | D4 | CE | 8A | B7 | 61 | 29 | 7F | 04 | 72 | B7 |
| 148 | B5 | 81 | 9 | 73 | C2 | 52 | FE | 08 | E4 | 73 |
| 149 | 77 | 1F | 12 | E6 | 99 | A4 | E1 | 10 | D5 | E6 |
| 150 | EE | 3E | 24 | D1 | 2F | 55 | DF | 20 | B7 | D1 |
| 151 | C1 | 7C | 48 | BF | 5E | AA | A3 | 40 | 73 | BF |
| 152 | 9F | F8 | 90 | 63 | BC | 49 | 5B | 80 | E6 | 63 |
| 153 | 23 | ED | 3D | C6 | 65 | 92 | B6 | 1D | D1 | C6 |
| 154 | 46 | C7 | 7A | 91 | CA | 39 | 71 | 3A | BF | 91 |
| 155 | 8C | 93 | F4 | 3F | 89 | 72 | E2 | 74 | 63 | 3F |
| 156 | 5 | 3B | F5 | 7E | F | E4 | D9 | E8 | C6 | 7E |
| 157 | A | 76 | F7 | FC | 1E | D5 | AF | CD | 91 | FC |
| 158 | 14 | EC | F3 | E5 | 3C | B7 | 43 | 87 | 3F | E5 |
| 159 | 28 | C5 | FB | D7 | 78 | 73 | 86 | 13 | 7E | D7 |
| 160 | 50 | 97 | EB | B3 | F0 | E6 | 11 | 26 | FC | B3 |
| 161 | A0 | 33 | CB | 7B | FD | D1 | 22 | 4C | E5 | 7B |
| 162 | 5D | 66 | 8B | F6 | E7 | BF | 44 | 98 | D7 | F6 |
| 163 | BA | CC | B | F1 | D3 | 63 | 88 | 2D | B3 | F1 |
| 164 | 69 | 85 | 16 | FF | BB | C6 | D | 5A | 7B | FF |
| 165 | D2 | 17 | 2C | E3 | 6B | 91 | 1A | B4 | F6 | E3 |
| 166 | B9 | 2E | 58 | DB | D6 | 3F | 34 | 75 | F1 | DB |
| 167 | 6F | 5C | B0 | AB | B1 | 7E | 68 | EA | FF | AB |
| 168 | DE | B8 | 7D | 4B | 7F | FC | D0 | C9 | E3 | 4B |
| 169 | A1 | 6D | FA | 96 | FE | E5 | BD | 8F | DB | 96 |
| 170 | 5F | DA | E9 | 31 | E1 | D7 | 67 | 3 | AB | 31 |
| 171 | BE | A9 | CF | 62 | DF | B3 | CE | 6 | 4B | 62 |
| 172 | 61 | 4F | 83 | C4 | A3 | 7B | 81 | C | 96 | C4 |

| 173 | C2 | 9E | 1B | 95 | 5B | F6 | 1F | 18 | 31 | 95 |
|---|---|---|---|---|---|---|---|---|---|---|
| 174 | 99 | 21 | 36 | 37 | B6 | F1 | 3E | 30 | 62 | 37 |
| 175 | 2F | 42 | 6C | 6E | 71 | FF | 7C | 60 | C4 | 6E |
| 176 | 5E | 84 | D8 | DC | E2 | E3 | F8 | C0 | 95 | DC |
| 177 | BC | 15 | AD | A5 | D9 | DB | ED | 9D | 37 | A5 |
| 178 | 65 | 2A | 47 | 57 | AF | AB | C7 | 27 | 6E | 57 |
| 179 | CA | 54 | 8E | AE | 43 | 4B | 93 | 4E | DC | AE |
| 180 | 89 | A8 | 02 | 41 | 86 | 96 | 3B | 9C | A5 | 41 |
| 181 | F | 4D | 04 | 82 | 11 | 31 | 76 | 25 | 57 | 82 |
| 182 | 1E | 9A | 08 | 19 | 22 | 62 | EC | 4A | AE | 19 |
| 183 | 3C | 29 | 10 | 32 | 44 | C4 | C5 | 94 | 41 | 32 |
| 184 | 78 | 52 | 20 | 64 | 88 | 95 | 97 | 35 | 82 | 64 |
| 185 | F0 | A4 | 40 | C8 | D | 37 | 33 | 6A | 19 | C8 |
| 186 | FD | 55 | 80 | 8D | 1A | 6E | 66 | D4 | 32 | 8D |
| 187 | E7 | AA | 1D | 7 | 34 | DC | CC | B5 | 64 | 7 |
| 188 | D3 | 49 | 3A | E | 68 | A5 | 85 | 77 | C8 | E |
| 189 | BB | 92 | 74 | 1C | D0 | 57 | 17 | EE | 8D | 1C |
| 190 | 6B | 39 | E8 | 38 | BD | AE | 2E | C1 | 7 | 38 |
| 191 | D6 | 72 | CD | 70 | 67 | 41 | 5C | 9F | E | 70 |
| 192 | B1 | E4 | 87 | E0 | CE | 82 | B8 | 23 | 1C | E0 |
| 193 | 7F | D5 | 13 | DD | 81 | 19 | 6D | 46 | 38 | DD |
| 194 | FE | B7 | 26 | A7 | 1F | 32 | DA | 8C | 70 | A7 |
| 195 | E1 | 73 | 4C | 53 | 3E | 64 | A9 | 5 | E0 | 53 |
| 196 | DF | E6 | 98 | A6 | 7C | C8 | 4F | A | DD | A6 |
| 197 | A3 | D1 | 2D | 51 | F8 | 8D | 9E | 14 | A7 | 51 |
| 198 | 5B | BF | 5A | A2 | ED | 7 | 21 | 28 | 53 | A2 |
| 199 | B6 | 63 | B4 | 59 | C7 | E | 42 | 50 | A6 | 59 |
| 200 | 71 | C6 | 75 | B2 | 93 | 1C | 84 | A0 | 51 | B2 |
| 201 | E2 | 91 | EA | 79 | 3B | 38 | 15 | 5D | A2 | 79 |
| 202 | D9 | 3F | C9 | F2 | 76 | 70 | 2A | BA | 59 | F2 |
| 203 | AF | 7E | 8F | F9 | EC | E0 | 54 | 69 | B2 | F9 |
| 204 | 43 | FC | 3 | EF | C5 | DD | A8 | D2 | 79 | EF |
| 205 | 86 | E5 | 6 | C3 | 97 | A7 | 4D | B9 | F2 | C3 |
| 206 | 11 | D7 | C | 9B | 33 | 53 | 9A | 6F | F9 | 9B |
| 207 | 22 | B3 | 18 | 2B | 66 | A6 | 29 | DE | EF | 2B |
| 208 | 44 | 7B | 30 | 56 | CC | 51 | 52 | A1 | C3 | 56 |
| 209 | 88 | F6 | 60 | AC | 85 | A2 | A4 | 5F | 9B | AC |
| 210 | D | F1 | C0 | 45 | 17 | 59 | 55 | BE | 2B | 45 |
| 211 | 1A | FF | 9D | 8A | 2E | B2 | AA | 61 | 56 | 8A |
| 212 | 34 | E3 | 27 | 9 | 5C | 79 | 49 | C2 | AC | 9 |
| 213 | 68 | DB | 4E | 12 | B8 | F2 | 92 | 99 | 45 | 12 |
| 214 | D0 | AB | 9C | 24 | 6D | F9 | 39 | 2F | 8A | 24 |
| 215 | BD | 4B | 25 | 48 | DA | EF | 72 | 5E | 9 | 48 |
| 216 | 67 | 96 | 4A | 90 | A9 | C3 | E4 | BC | 12 | 90 |
| 217 | CE | 31 | 94 | 3D | 4F | 9B | D5 | 65 | 24 | 3D |

| 218 | 81 | 62 | 35 | 7A | 9E | 2B | B7 | CA | 48 | 7A |
|---|---|---|---|---|---|---|---|---|---|---|
| 219 | 1F | C4 | 6A | F4 | 21 | 56 | 73 | 89 | 90 | F4 |
| 220 | 3E | 95 | D4 | F5 | 42 | AC | E6 | F | 3D | F5 |
| 221 | 7C | 37 | B5 | F7 | 84 | 45 | D1 | 1E | 7A | F7 |
| 222 | F8 | 6E | 77 | F3 | 15 | 8A | BF | 3C | F4 | F3 |
| 223 | ED | DC | EE | FB | 2A | 9 | 63 | 78 | F5 | FB |
| 224 | C7 | A5 | C1 | EB | 54 | 12 | C6 | F0 | F7 | EB |
| 225 | 93 | 57 | 9F | CB | A8 | 24 | 91 | FD | F3 | CB |
| 226 | 3B | AE | 23 | 8B | 4D | 48 | 3F | E7 | FB | 8B |
| 227 | 76 | 41 | 46 | B | 9A | 90 | 7E | D3 | EB | B |
| 228 | EC | 82 | 8C | 16 | 29 | 3D | FC | BB | CB | 16 |
| 229 | C5 | 19 | 5 | 2C | 52 | 7A | E5 | 6B | 8B | 2C |
| 230 | 97 | 32 | A | 58 | A4 | F4 | D7 | D6 | B | 58 |
| 231 | 33 | 64 | 14 | B0 | 55 | F5 | B3 | B1 | 16 | B0 |
| 232 | 66 | C8 | 28 | 7D | AA | F7 | 7B | 7F | 2C | 7D |
| 233 | CC | 8D | 50 | FA | 49 | F3 | F6 | FE | 58 | FA |
| 234 | 85 | 7 | A0 | E9 | 92 | FB | F1 | E1 | B0 | E9 |
| 235 | 17 | E | 5D | CF | 39 | EB | FF | DF | 7D | CF |
| 236 | 2E | 1C | BA | 83 | 72 | CB | E3 | A3 | FA | 83 |
| 237 | 5C | 38 | 69 | 1B | E4 | 8B | DB | 5B | E9 | 1B |
| 238 | B8 | 70 | D2 | 36 | D5 | B | AB | B6 | CF | 36 |
| 239 | 6D | E0 | B9 | 6C | B7 | 16 | 4B | 71 | 83 | 6C |
| 240 | DA | DD | 6F | D8 | 73 | 2C | 96 | E2 | 1B | D8 |
| 241 | A9 | A7 | DE | AD | E6 | 58 | 31 | D9 | 36 | AD |
| 242 | 4F | 53 | A1 | 47 | D1 | B0 | 62 | AF | 6C | 47 |
| 243 | 9E | A6 | 5F | 8E | BF | 7D | C4 | 43 | D8 | 8E |
| 244 | 21 | 51 | BE | 02 | 63 | FA | 95 | 86 | AD | 02 |
| 245 | 42 | A2 | 61 | 04 | C6 | E9 | 37 | 11 | 47 | 04 |
| 246 | 84 | 59 | C2 | 08 | 91 | CF | 6E | 22 | 8E | 08 |
| 247 | 15 | B2 | 99 | 10 | 3F | 83 | DC | 44 | 02 | 10 |
| 248 | 2A | 79 | 2F | 20 | 7E | 1B | A5 | 88 | 04 | 20 |
| 249 | 54 | F2 | 5E | 40 | FC | 36 | 57 | D | 08 | 40 |
| 250 | A8 | F9 | BC | 80 | E5 | 6C | AE | 1A | 10 | 80 |
| 251 | 4D | EF | 65 | 1D | D7 | D8 | 41 | 34 | 20 | 1D |
| 252 | 9A | C3 | CA | 3A | B3 | AD | 82 | 68 | 40 | 3A |
| 253 | 29 | 9B | 89 | 74 | 7B | 47 | 19 | D0 | 80 | 74 |
| 254 | 52 | 2B | F | E8 | F6 | 8E | 32 | BD | 1D | E8 |

Table B1. The ROMs Tables

## Appendix C: The output of the C code

- The generator polynomial

| | |
|---|---|
| alpha_to[ 1]= 2 | gp gg[ 0 ]= 45 |
| alpha_to[ 2]= 4 | gp gg[ 1 ]=216 |
| alpha_to[ 3]= 8 | gp gg[ 2 ]=239 |
| alpha_to[ 4]= 16 | gp gg[ 3 ]= 24 |
| alpha_to[ 5]= 32 | gp gg[ 4 ]=253 |
| alpha_to[ 6]= 64 | gp gg[ 5 ]=104 |
| alpha_to[ 7]=128 | gp gg[ 6 ]= 27 |
| alpha_to[ 8]= 29 | gp gg[ 7 ]= 40 |
| alpha_to[ 9]= 58 | gp gg[ 8 ]=107 |
| alpha_to[10]=116 | gp gg[ 9 ]= 50 |
| alpha_to[11]=232 | gp gg[10 ]=163 |
| alpha_to[12]=205 | gp gg[11 ]=210 |
| alpha_to[13]=135 | gp gg[12 ]=227 |
| alpha_to[14]= 19 | gp gg[13 ]=134 |
| alpha_to[15]= 38 | gp gg[14 ]=224 |
| alpha_to[16]= 76 | gp gg[15 ]=158 |
| alpha_to[17]=152 | gp gg[16 ]=119 |
| alpha_to[18]= 45 | gp gg[17 ]= 13 |
| alpha_to[19]= 90 | gp gg[18 ]=158 |
| alpha_to[20]=180 | gp gg[19 ]= 1 |
| alpha_to[21]=117 | gp gg[20 ]=238 |
| alpha_to[22]=234 | gp gg[21 ]=164 |
| alpha_to[23]=201 | gp gg[22 ]= 82 |
| alpha_to[24]=143 | gp gg[23 ]= 43 |
| alpha_to[25]= 3 | gp gg[24 ]= 15 |
| alpha_to[26]= 6 | gp gg[25 ]=232 |
| alpha_to[27]= 12 | gp gg[26 ]=246 |
| alpha_to[28]= 24 | gp gg[27 ]=142 |
| alpha_to[29]= 48 | gp gg[28 ]= 50 |
| alpha_to[30]= 96 | gp gg[29 ]=189 |
| alpha_to[31]=192 | gp gg[30 ]= 29 |
| alpha_to[32]=157 | gp gg[31 ]=232 |

- Look-up tables for GF (2**8)

| i in DEC Form | i in HEX Form | alpha_to[i] | index_of[i] in HEX Form | index_of[i] in DEC Form |
|---|---|---|---|---|
| 0 | 0 | 1 | FFFFFFFF | -1 |
| 1 | 1 | 2 | 0 | 0 |
| 2 | 2 | 4 | 1 | 1 |
| 3 | 3 | 8 | 19 | 25 |
| 4 | 4 | 10 | 2 | 2 |
| 5 | 5 | 20 | 32 | 50 |
| 6 | 6 | 40 | 1A | 26 |
| 7 | 7 | 80 | C6 | 198 |
| 8 | 8 | 1D | 3 | 3 |
| 9 | 9 | 3A | DF | 223 |
| 10 | A | 74 | 33 | 51 |
| 11 | B | E8 | EE | 238 |
| 12 | C | CD | 1B | 27 |
| 13 | D | 87 | 68 | 104 |
| 14 | E | 13 | C7 | 199 |
| 15 | F | 26 | 4B | 75 |
| 16 | 10 | 4C | 4 | 4 |
| 17 | 11 | 98 | 64 | 100 |
| 18 | 12 | 2D | E0 | 224 |
| 19 | 13 | 5A | E | 14 |
| 20 | 14 | B4 | 34 | 52 |
| 21 | 15 | 75 | 8D | 141 |
| 22 | 16 | EA | EF | 239 |
| 23 | 17 | C9 | 81 | 129 |
| 24 | 18 | 8F | 1C | 28 |
| 25 | 19 | 3 | C1 | 193 |
| 26 | 1A | 6 | 69 | 105 |
| 27 | 1B | C | F8 | 248 |
| 28 | 1C | 18 | C8 | 200 |
| 29 | 1D | 30 | 8 | 8 |
| 30 | 1E | 60 | 4C | 76 |
| 31 | 1F | C0 | 71 | 113 |
| 32 | 20 | 9D | 5 | 5 |
| 33 | 21 | 27 | 8A | 138 |
| 34 | 22 | 4E | 65 | 101 |
| 35 | 23 | 9C | 2F | 47 |
| 36 | 24 | 25 | E1 | 225 |
| 37 | 25 | 4A | 24 | 36 |
| 38 | 26 | 94 | F | 15 |

| | | | | |
|---|---|---|---|---|
| 39 | 27 | 35 | 21 | 33 |
| 40 | 28 | 6A | 35 | 53 |
| 41 | 29 | D4 | 93 | 147 |
| 42 | 2A | B5 | 8E | 142 |
| 43 | 2B | 77 | DA | 218 |
| 44 | 2C | EE | F0 | 240 |
| 45 | 2D | C1 | 12 | 18 |
| 46 | 2E | 9F | 82 | 130 |
| 47 | 2F | 23 | 45 | 69 |
| 48 | 30 | 46 | 1D | 29 |
| 49 | 31 | 8C | B5 | 181 |
| 50 | 32 | 5 | C2 | 194 |
| 51 | 33 | A | 7D | 125 |
| 52 | 34 | 14 | 6A | 106 |
| 53 | 35 | 28 | 27 | 39 |
| 54 | 36 | 50 | F9 | 249 |
| 55 | 37 | A0 | B9 | 185 |
| 56 | 38 | 5D | C9 | 201 |
| 57 | 39 | BA | 9A | 154 |
| 58 | 3A | 69 | 9 | 9 |
| 59 | 3B | D2 | 78 | 120 |
| 60 | 3C | B9 | 4D | 77 |
| 61 | 3D | 6F | E4 | 228 |
| 62 | 3E | DE | 72 | 114 |
| 63 | 3F | A1 | A6 | 166 |
| 64 | 40 | 5F | 6 | 6 |
| 65 | 41 | BE | BF | 191 |
| 66 | 42 | 61 | 8B | 139 |
| 67 | 43 | C2 | 62 | 98 |
| 68 | 44 | 99 | 66 | 102 |
| 69 | 45 | 2F | DD | 221 |
| 70 | 46 | 5E | 30 | 48 |
| 71 | 47 | BC | FD | 253 |
| 72 | 48 | 65 | E2 | 226 |
| 73 | 49 | CA | 98 | 152 |
| 74 | 4A | 89 | 25 | 37 |
| 75 | 4B | F | B3 | 179 |
| 76 | 4C | 1E | 10 | 16 |
| 77 | 4D | 3C | 91 | 145 |
| 78 | 4E | 78 | 22 | 34 |
| 79 | 4F | F0 | 88 | 136 |
| 80 | 50 | FD | 36 | 54 |
| 81 | 51 | E7 | D0 | 208 |
| 82 | 52 | D3 | 94 | 148 |
| 83 | 53 | BB | CE | 206 |

| | | | | |
|---|---|---|---|---|
| 84 | 54 | 6B | 8F | 143 |
| 85 | 55 | D6 | 96 | 150 |
| 86 | 56 | B1 | DB | 219 |
| 87 | 57 | 7F | BD | 189 |
| 88 | 58 | FE | F1 | 241 |
| 89 | 59 | E1 | D2 | 210 |
| 90 | 5A | DF | 13 | 19 |
| 91 | 5B | A3 | 5C | 92 |
| 92 | 5C | 5B | 83 | 131 |
| 93 | 5D | B6 | 38 | 56 |
| 94 | 5E | 71 | 46 | 70 |
| 95 | 5F | E2 | 40 | 64 |
| 96 | 60 | D9 | 1E | 30 |
| 97 | 61 | AF | 42 | 66 |
| 98 | 62 | 43 | B6 | 182 |
| 99 | 63 | 86 | A3 | 163 |
| 100 | 64 | 11 | C3 | 195 |
| 101 | 65 | 22 | 48 | 72 |
| 102 | 66 | 44 | 7E | 126 |
| 103 | 67 | 88 | 6E | 110 |
| 104 | 68 | D | 6B | 107 |
| 105 | 69 | 1A | 3A | 58 |
| 106 | 6A | 34 | 28 | 40 |
| 107 | 6B | 68 | 54 | 84 |
| 108 | 6C | D0 | FA | 250 |
| 109 | 6D | BD | 85 | 133 |
| 110 | 6E | 67 | BA | 186 |
| 111 | 6F | CE | 3D | 61 |
| 112 | 70 | 81 | CA | 202 |
| 113 | 71 | 1F | 5E | 94 |
| 114 | 72 | 3E | 9B | 155 |
| 115 | 73 | 7C | 9F | 159 |
| 116 | 74 | F8 | A | 10 |
| 117 | 75 | ED | 15 | 21 |
| 118 | 76 | C7 | 79 | 121 |
| 119 | 77 | 93 | 2B | 43 |
| 120 | 78 | 3B | 4E | 78 |
| 121 | 79 | 76 | D4 | 212 |
| 122 | 7A | EC | E5 | 229 |
| 123 | 7B | C5 | AC | 172 |
| 124 | 7C | 97 | 73 | 115 |
| 125 | 7D | 33 | F3 | 243 |
| 126 | 7E | 66 | A7 | 167 |
| 127 | 7F | CC | 57 | 87 |
| 128 | 80 | 85 | 7 | 7 |

| | | | | |
|---|---|---|---|---|
| 129 | 81 | 17 | 70 | 112 |
| 130 | 82 | 2E | C0 | 192 |
| 131 | 83 | 5C | F7 | 247 |
| 132 | 84 | B8 | 8C | 140 |
| 133 | 85 | 6D | 80 | 128 |
| 134 | 86 | DA | 63 | 99 |
| 135 | 87 | A9 | D | 13 |
| 136 | 88 | 4F | 67 | 103 |
| 137 | 89 | 9E | 4A | 74 |
| 138 | 8A | 21 | DE | 222 |
| 139 | 8B | 42 | ED | 237 |
| 140 | 8C | 84 | 31 | 49 |
| 141 | 8D | 15 | C5 | 197 |
| 142 | 8E | 2A | FE | 254 |
| 143 | 8F | 54 | 18 | 24 |
| 144 | 90 | A8 | E3 | 227 |
| 145 | 91 | 4D | A5 | 165 |
| 146 | 92 | 9A | 99 | 153 |
| 147 | 93 | 29 | 77 | 119 |
| 148 | 94 | 52 | 26 | 38 |
| 149 | 95 | A4 | B8 | 184 |
| 150 | 96 | 55 | B4 | 180 |
| 151 | 97 | AA | 7C | 124 |
| 152 | 98 | 49 | 11 | 17 |
| 153 | 99 | 92 | 44 | 68 |
| 154 | 9A | 39 | 92 | 146 |
| 155 | 9B | 72 | D9 | 217 |
| 156 | 9C | E4 | 23 | 35 |
| 157 | 9D | D5 | 20 | 32 |
| 158 | 9E | B7 | 89 | 137 |
| 159 | 9F | 73 | 2E | 46 |
| 160 | A0 | E6 | 37 | 55 |
| 161 | A1 | D1 | 3F | 63 |
| 162 | A2 | BF | D1 | 209 |
| 163 | A3 | 63 | 5B | 91 |
| 164 | A4 | C6 | 95 | 149 |
| 165 | A5 | 91 | BC | 188 |
| 166 | A6 | 3F | CF | 207 |
| 167 | A7 | 7E | CD | 205 |
| 168 | A8 | FC | 90 | 144 |
| 169 | A9 | E5 | 87 | 135 |
| 170 | AA | D7 | 97 | 151 |
| 171 | AB | B3 | B2 | 178 |
| 172 | AC | 7B | DC | 220 |
| 173 | AD | F6 | FC | 252 |

| | | | | |
|---|---|---|---|---|
| 174 | AE | F1 | BE | 190 |
| 175 | AF | FF | 61 | 97 |
| 176 | B0 | E3 | F2 | 242 |
| 177 | B1 | DB | 56 | 86 |
| 178 | B2 | AB | D3 | 211 |
| 179 | B3 | 4B | AB | 171 |
| 180 | B4 | 96 | 14 | 20 |
| 181 | B5 | 31 | 2A | 42 |
| 182 | B6 | 62 | 5D | 93 |
| 183 | B7 | C4 | 9E | 158 |
| 184 | B8 | 95 | 84 | 132 |
| 185 | B9 | 37 | 3C | 60 |
| 186 | BA | 6E | 39 | 57 |
| 187 | BB | DC | 53 | 83 |
| 188 | BC | A5 | 47 | 71 |
| 189 | BD | 57 | 6D | 109 |
| 190 | BE | AE | 41 | 65 |
| 191 | BF | 41 | A2 | 162 |
| 192 | C0 | 82 | 1F | 31 |
| 193 | C1 | 19 | 2D | 45 |
| 194 | C2 | 32 | 43 | 67 |
| 195 | C3 | 64 | D8 | 216 |
| 196 | C4 | C8 | B7 | 183 |
| 197 | C5 | 8D | 7B | 123 |
| 198 | C6 | 7 | A4 | 164 |
| 199 | C7 | E | 76 | 118 |
| 200 | C8 | 1C | C4 | 196 |
| 201 | C9 | 38 | 17 | 23 |
| 202 | CA | 70 | 49 | 73 |
| 203 | CB | E0 | EC | 236 |
| 204 | CC | DD | 7F | 127 |
| 205 | CD | A7 | C | 12 |
| 206 | CE | 53 | 6F | 111 |
| 207 | CF | A6 | F6 | 246 |
| 208 | D0 | 51 | 6C | 108 |
| 209 | D1 | A2 | A1 | 161 |
| 210 | D2 | 59 | 3B | 59 |
| 211 | D3 | B2 | 52 | 82 |
| 212 | D4 | 79 | 29 | 41 |
| 213 | D5 | F2 | 9D | 157 |
| 214 | D6 | F9 | 55 | 85 |
| 215 | D7 | EF | AA | 170 |
| 216 | D8 | C3 | FB | 251 |
| 217 | D9 | 9B | 60 | 96 |
| 218 | DA | 2B | 86 | 134 |

| | | | | |
|---|---|---|---|---|
| 219 | DB | 56 | B1 | 177 |
| 220 | DC | AC | BB | 187 |
| 221 | DD | 45 | CC | 204 |
| 222 | DE | 8A | 3E | 62 |
| 223 | DF | 9 | 5A | 90 |
| 224 | E0 | 12 | CB | 203 |
| 225 | E1 | 24 | 59 | 89 |
| 226 | E2 | 48 | 5F | 95 |
| 227 | E3 | 90 | B0 | 176 |
| 228 | E4 | 3D | 9C | 156 |
| 229 | E5 | 7A | A9 | 169 |
| 230 | E6 | F4 | A0 | 160 |
| 231 | E7 | F5 | 51 | 81 |
| 232 | E8 | F7 | B | 11 |
| 233 | E9 | F3 | F5 | 245 |
| 234 | EA | FB | 16 | 22 |
| 235 | EB | EB | EB | 235 |
| 236 | EC | CB | 7A | 122 |
| 237 | ED | 8B | 75 | 117 |
| 238 | EE | B | 2C | 44 |
| 239 | EF | 16 | D7 | 215 |
| 240 | F0 | 2C | 4F | 79 |
| 241 | F1 | 58 | AE | 174 |
| 242 | F2 | B0 | D5 | 213 |
| 243 | F3 | 7D | E9 | 233 |
| 244 | F4 | FA | E6 | 230 |
| 245 | F5 | E9 | E7 | 231 |
| 246 | F6 | CF | AD | 173 |
| 247 | F7 | 83 | E8 | 232 |
| 248 | F8 | 1B | 74 | 116 |
| 249 | F9 | 36 | D6 | 214 |
| 250 | FA | 6C | F4 | 244 |
| 251 | FB | D8 | EA | 234 |
| 252 | FC | AD | A8 | 168 |
| 253 | FD | 47 | 50 | 80 |
| 254 | FE | 8E | 58 | 88 |

Results for Reed-Solomon code (n = 255, k = 223, t = 16) (data, recd in polynomial form)

The first 32 is the parity check digits and the rest are the massage 223 to form 255 codeword

| i | data[i] | recd[i](decoded) |
|---|---|---|
| 0 | 120 | 120 |
| 1 | 41 | 41 |
| 2 | 190 | 190 |
| 3 | 87 | 87 |
| 4 | 41 | 41 |
| 5 | 197 | 197 |
| 6 | 214 | 214 |
| 7 | 196 | 196 |
| 8 | 192 | 192 |
| 9 | 17 | 17 |
| 10 | 239 | 239 |
| 11 | 31 | 31 |
| 12 | 208 | 208 |
| 13 | 221 | 221 |
| 14 | 2 | 2 |
| 15 | 196 | 196 |
| 16 | 251 | 251 |
| 17 | 31 | 31 |
| 18 | 17 | 17 |
| 19 | 171 | 171 |
| 20 | 240 | 240 |
| 21 | 225 | 225 |
| 22 | 164 | 164 |
| 23 | 61 | 61 |
| 24 | 184 | 184 |
| 25 | 155 | 155 |
| 26 | 22 | 22 |
| 27 | 239 | 239 |
| 28 | 17 | 17 |
| 29 | 65 | 65 |
| 30 | 237 | 237 |
| 31 | 104 | 104 |
| 32 | 223 | 223 |
| 33 | 222 | 222 |
| 34 | 221 | 221 |
| 35 | 220 | 220 |
| 36 | 219 | 219 |
| 37 | 218 | 218 |

| | | |
|---|---|---|
| 38 | 217 | 217 |
| 39 | 216 | 216 |
| 40 | 215 | 215 |
| 41 | 214 | 214 |
| 42 | 213 | 213 |
| 43 | 212 | 212 |
| 44 | 211 | 211 |
| 45 | 210 | 210 |
| 46 | 209 | 209 |
| 47 | 208 | 208 |
| 48 | 207 | 207 |
| 49 | 206 | 206 |
| 50 | 205 | 205 |
| 51 | 204 | 204 |
| 52 | 203 | 203 |
| 53 | 202 | 202 |
| 54 | 201 | 201 |
| 55 | 200 | 200 |
| 56 | 199 | 199 |
| 57 | 198 | 198 |
| 58 | 197 | 197 |
| 59 | 196 | 196 |
| 60 | 195 | 195 |
| 61 | 194 | 194 |
| 62 | 193 | 193 |
| 63 | 192 | 192 |
| 64 | 191 | 191 |
| 65 | 190 | 190 |
| 66 | 189 | 189 |
| 67 | 188 | 188 |
| 68 | 187 | 187 |
| 69 | 186 | 186 |
| 70 | 185 | 185 |
| 71 | 184 | 184 |
| 72 | 183 | 183 |
| 73 | 182 | 182 |
| 74 | 181 | 181 |
| 75 | 180 | 180 |
| 76 | 179 | 179 |
| 77 | 178 | 178 |
| 78 | 177 | 177 |
| 79 | 176 | 176 |
| 80 | 175 | 175 |
| 81 | 174 | 174 |
| 82 | 173 | 173 |

| | | |
|---|---|---|
| 83 | 172 | 172 |
| 84 | 171 | 171 |
| 85 | 170 | 170 |
| 86 | 169 | 169 |
| 87 | 168 | 168 |
| 88 | 167 | 167 |
| 89 | 166 | 166 |
| 90 | 165 | 165 |
| 91 | 164 | 164 |
| 92 | 163 | 163 |
| 93 | 162 | 162 |
| 94 | 161 | 161 |
| 95 | 160 | 160 |
| 96 | 159 | 159 |
| 97 | 158 | 158 |
| 98 | 157 | 157 |
| 99 | 156 | 156 |
| 100 | 155 | 155 |
| 101 | 154 | 154 |
| 102 | 153 | 153 |
| 103 | 152 | 152 |
| 104 | 151 | 151 |
| 105 | 150 | 150 |
| 106 | 149 | 149 |
| 107 | 148 | 148 |
| 108 | 147 | 147 |
| 109 | 146 | 146 |
| 110 | 145 | 145 |
| 111 | 144 | 144 |
| 112 | 143 | 143 |
| 113 | 142 | 142 |
| 114 | 141 | 141 |
| 115 | 140 | 140 |
| 116 | 139 | 139 |
| 117 | 138 | 138 |
| 118 | 137 | 137 |
| 119 | 136 | 136 |
| 120 | 135 | 135 |
| 121 | 134 | 134 |
| 122 | 133 | 133 |
| 123 | 132 | 132 |
| 124 | 131 | 131 |
| 125 | 130 | 130 |
| 126 | 129 | 129 |
| 127 | 128 | 128 |

| | | |
|---|---|---|
| 128 | 127 | 127 |
| 129 | 126 | 126 |
| 130 | 125 | 125 |
| 131 | 124 | 124 |
| 132 | 123 | 123 |
| 133 | 122 | 122 |
| 134 | 121 | 121 |
| 135 | 120 | 120 |
| 136 | 119 | 119 |
| 137 | 118 | 118 |
| 138 | 117 | 117 |
| 139 | 116 | 116 |
| 140 | 115 | 115 |
| 141 | 114 | 114 |
| 142 | 113 | 113 |
| 143 | 112 | 112 |
| 144 | 111 | 111 |
| 145 | 110 | 110 |
| 146 | 109 | 109 |
| 147 | 108 | 108 |
| 148 | 107 | 107 |
| 149 | 106 | 106 |
| 150 | 105 | 105 |
| 151 | 104 | 104 |
| 152 | 103 | 103 |
| 153 | 102 | 102 |
| 154 | 101 | 101 |
| 155 | 100 | 100 |
| 156 | 99 | 99 |
| 157 | 98 | 98 |
| 158 | 97 | 97 |
| 159 | 96 | 96 |
| 160 | 16 | 95 |
| 161 | 94 | 94 |
| 162 | 93 | 93 |
| 163 | 92 | 92 |
| 164 | 91 | 91 |
| 165 | 90 | 90 |
| 166 | 89 | 89 |
| 167 | 88 | 88 |
| 168 | 87 | 87 |
| 169 | 86 | 86 |
| 170 | 85 | 85 |
| 171 | 84 | 84 |
| 172 | 83 | 83 |

| | | |
|---|---|---|
| 173 | 82 | 82 |
| 174 | 81 | 81 |
| 175 | 80 | 80 |
| 176 | 79 | 79 |
| 177 | 78 | 78 |
| 178 | 77 | 77 |
| 179 | 76 | 76 |
| 180 | 75 | 75 |
| 181 | 74 | 74 |
| 182 | 73 | 73 |
| 183 | 72 | 72 |
| 184 | 71 | 71 |
| 185 | 70 | 70 |
| 186 | 69 | 69 |
| 187 | 68 | 68 |
| 188 | 67 | 67 |
| 189 | 66 | 66 |
| 190 | 65 | 65 |
| 191 | 64 | 64 |
| 192 | 63 | 63 |
| 193 | 62 | 62 |
| 194 | 61 | 61 |
| 195 | 60 | 60 |
| 196 | 59 | 59 |
| 197 | 58 | 58 |
| 198 | 57 | 57 |
| 199 | 56 | 56 |
| 200 | 55 | 55 |
| 201 | 54 | 54 |
| 202 | 53 | 53 |
| 203 | 52 | 52 |
| 204 | 51 | 51 |
| 205 | 50 | 50 |
| 206 | 49 | 49 |
| 207 | 48 | 48 |
| 208 | 47 | 47 |
| 209 | 46 | 46 |
| 210 | 45 | 45 |
| 211 | 44 | 44 |
| 212 | 43 | 43 |
| 213 | 42 | 42 |
| 214 | 41 | 41 |
| 215 | 40 | 40 |
| 216 | 39 | 39 |
| 217 | 38 | 38 |

| | | |
|---|---|---|
| 218 | 37 | 37 |
| 219 | 36 | 36 |
| 220 | 35 | 35 |
| 221 | 34 | 34 |
| 222 | 33 | 33 |
| 223 | 32 | 32 |
| 224 | 31 | 31 |
| 225 | 30 | 30 |
| 226 | 29 | 29 |
| 227 | 28 | 28 |
| 228 | 27 | 27 |
| 229 | 26 | 26 |
| 230 | 25 | 25 |
| 231 | 24 | 24 |
| 232 | 23 | 23 |
| 233 | 22 | 22 |
| 234 | 21 | 21 |
| 235 | 20 | 20 |
| 236 | 19 | 19 |
| 237 | 18 | 18 |
| 238 | 17 | 17 |
| 239 | 16 | 16 |
| 240 | 15 | 15 |
| 241 | 14 | 14 |
| 242 | 13 | 13 |
| 243 | 12 | 12 |
| 244 | 11 | 11 |
| 245 | 10 | 10 |
| 246 | 9 | 9 |
| 247 | 8 | 8 |
| 248 | 7 | 7 |
| 249 | 6 | 6 |
| 250 | 5 | 5 |
| 251 | 4 | 4 |
| 252 | 3 | 3 |
| 253 | 2 | 2 |
| 254 | 1 | 1 |