GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Project Information theory

# Compact Disc Digital Audio

Thomas Delva
Simon Van den Eynde
Dries Van Leemput

Prof. dr. ir. Heidi Steendam
Ir. Johannes Van Wonterghem

Academic year 2017-2018

# Contents

# 1 Introduction

This work is a university project for the course information theory ($E003600$). The assignment can be found in the file `Assignment.pdf` accompanying this document.

In the first section we will answer some question regarding the technical aspects of the code structure and data transmission on CD's.

In section 2 we will answer some theoretical questions on Reed-Solomon encoding and decoding. Furthermore we programmed a Reed-Solomon encoder and decoder in Matlab, see the file `RSCode.m`.

In section 3 we implemented the CD-standard Matlab encoding, by using built-in function. See the file `AudioCD.m` Thereby we also answered some questions on how the algorithm works and how it performs.

# 2 Audio CD Subcode

In this first section, the audio CD subcode is examined. In each audio CD frame, 8 subcode bits (1 symbol) are included apart from the audio samples and 27 sync bits. Those bits are available for control and display purpose. The datarate and the two used channels of the subcode: P-channel and Q-channel are handled.

## 2.1 Datarate

The subcode bits are transmitted in blocks of 98 symbols with a repetition frequency of 75 Hz. The datarate in subcode blocks/s is therefore 0.0133 subcode blocks/s ($\frac{1}{75\,\mathrm{Hz}}$). To compute the datarate in bits/s, first the content of a subcode block has to be examined. Each block starts with two subcoding sync patterns of 14 bit. Next, 96 8 bit subcode symbols are transmitted as 14 bit because they are EFM-coded (Eight to Fourteen Modulation). It follows that the total amount of bits in one subcode block is 1372 ($14\,\mathrm{bit} * 98$), so the datarate in bit/s is 18.29 bit/s ($\frac{1372\,\mathrm{bit}}{75\,\mathrm{Hz}}$).

## 2.2 P-channel

One of the subcode bits represents the P channel flag bit. This flag indicates the start of a track. If the P-flag is 0, the track is running. 1 indicates the start of a track. The minimum duration of the flag is 2 seconds. If the pause between tracks exceeds 2 seconds, the flag will be high during the time of pause. At the end of the last track, the flag will be high for 2-3 seconds until lead-out. After lead-out, the flag switches between 0 and 1 with a frequency of 2 Hz after an idle time of again 2-3 seconds. The state of the P channel flag under various circumstances is depicted in figure 1.

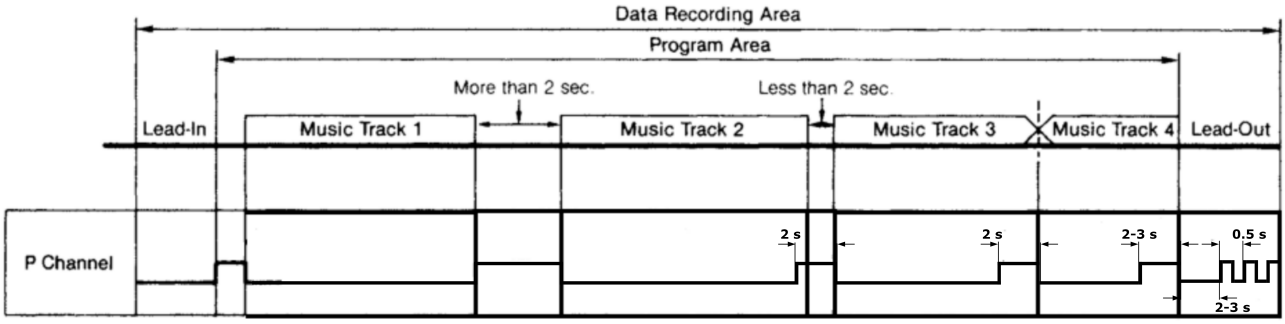


Figure 1: State of P channel flag

## 2.3 Q-channel

The CD player obtains information about the CD during lead-in, using the Q-channel of the subcode. The number of tracks, the starting positions of those tracks and the total playing time of the audio CD can be retrieved. Figure 2 shows the structure of a Q-channel. As expected, 96 bits are transmitted since a subcode block consists of 98 symbols, where two are reserved for synchronization. The first 4 control bits determine whether or not pre-emphasis is used and if copy is permitted or prohibited. The next 4 address bits determine the DATA-Q mode. During lead-in, the first mode is used to transmit the table of contents, corresponding to an address field of 0000.

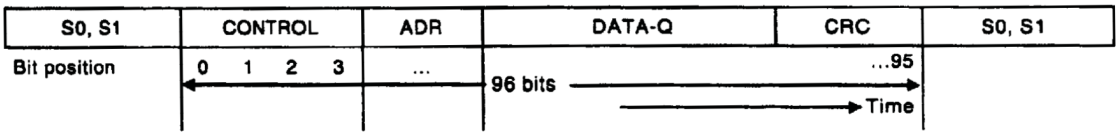


Figure 2: Structure of Q-channel

The data structure of the Q-channel is shown in figure 3. TNO represents the track number and remains fixed on 00 during lead-in. POINT, P MIN, P SEC and P FRAME are used to transmit the table of contents. POINT represents the track number. P MIN (minutes), P SEC (seconds) and P FRAME (frames, 75 fps) give the starting point of the track by an accuracy of $\pm$ 1 s. The actual start position is the first position with a new TNO and a nonzero index relative to TNO (which is only used in the Q-channel during audio and lead-out). Each frame is terminated with a 16 bit CRC to protect the control data from errors.

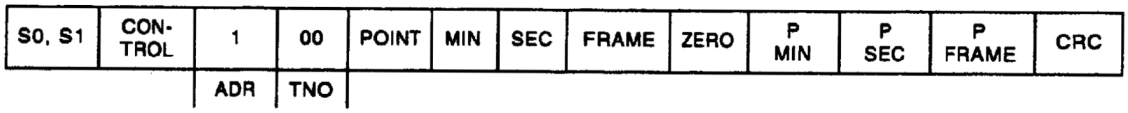


Figure 3: Data structure of Q-channel

During lead-in, the items in the table of contents are repeated three times. This also contributes to protection from errors. The TNO of the first three frames will be 01, the next 3 01 and so on until TNO reaches AA, which corresponds to the lead-out. The CD players therefore also knows the total playing time of the audio CD. After AA, the table of contents will be repeated in the same way until the first track begins.

# 3 Reed-Solomon code

## 3.1 Encoding

**Q1** We first need to check the irreducibility of $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ over $GF(2)$. We will do this by generating all polynomials of degree at most 4 in $GF(2)$. So we get, without the 0 polynomial, $2^5 - 1 = 31$ polynomials: $1, x, x+1 \dots x^4 + x^3 + x + 1$. In Matlab we can divide p(x) by these polynomials. We find that the remainder is never 0. So $p(x)$ can not be factored using only elements in $GF(2)$, thus $p(x)$ is irreducible.
Now let $\alpha$ be a primitive element of $GF(2^8)$. If $\alpha$ is a root of $p(x)$, then the elements $\alpha^{2^i}$ for $i \leq 7$ are roots as well. Then we can calculate

$$\begin{aligned}\prod_{i=0}^{i=7}(x - \alpha^{2^i}) &= (x-\alpha)(x-\alpha^2)(x-\alpha^4)\dots(x-\alpha^{128}) \\ &= x^8 + x^4 + x^3 + x^2 + 1\end{aligned}$$

So $\alpha$ is a root of $p(x)$.

**Q2** The $C_1$ Reed-Solomon code is an $(32, 28)$-code over $GF(2^8)$ modulo $P(X) := x^8 + x^4 + x^3 + x^2 + 1$. For the remaining questions in this section, we will use $n = 32$ and $k = 28$. A primitive element $\alpha$ is $[0\,0\,0\,0\,0\,0\,1\,0]$. We first construct the table of elements in $GF(2^8)$ using $P(X)$. The first elements are:

$$\begin{aligned}\alpha^0 &= [0\,0\,0\,0\,0\,0\,0\,1] & \alpha^6 &= [0\,1\,0\,0\,0\,0\,0\,0] \\ \alpha^1 &= [0\,0\,0\,0\,0\,0\,1\,0] & \alpha^7 &= [1\,0\,0\,0\,0\,0\,0\,0] \\ \alpha^2 &= [0\,0\,0\,0\,0\,1\,0\,0] & \alpha^8 &= [0\,0\,0\,1\,1\,1\,0\,1] \\ \alpha^3 &= [0\,0\,0\,0\,1\,0\,0\,0] & \alpha^9 &= [0\,0\,1\,1\,1\,0\,1\,0] \\ \alpha^4 &= [0\,0\,0\,1\,0\,0\,0\,0] & \alpha^{10} &= [0\,1\,1\,1\,0\,1\,0\,0] \\ \alpha^5 &= [0\,0\,1\,0\,0\,0\,0\,0] & \alpha^{11} &= [1\,1\,1\,0\,1\,0\,0\,0]\end{aligned}$$

The generator polynomial of Reed-Solomon code is of the form $g(x) = (x-\alpha)(x-\alpha^2)\dots(x-\alpha^{2t})$ where $2t$ is the number of parity symbols, which is $n - k = 4$. So

$$\begin{aligned}g(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) \\ &= (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4) \\ &= x^4 + (\alpha + \alpha^2 + \alpha^3 + \alpha^4)x^3 + (\alpha^3 + \alpha^4 + 2\alpha^5 + \alpha^6 + \alpha^7)x^2 + (\alpha^6 + \alpha^7 + \alpha^8 + \alpha^9)x + \alpha^{10} \\ &= x^4 + \alpha^{76}x^3 + \alpha^{251}x^2 + \alpha^{81} + \alpha^{10}\end{aligned}$$

Where the last transition was calculated with the full table of elements in $GF(2^8)$, for example: $\alpha^{78} = [0\,0\,0\,1\,1\,1\,1\,0]$. So $g_0 = \alpha^{10}$, $g_1 = \alpha^{81}$, $g_2 = \alpha^{251}$, $g_3 = \alpha^{76}$ and $g_4 = \alpha^0$.

We constructed $g(x)$ from its roots: $\alpha^1, \alpha^2, \alpha^3, \alpha^4$. Therefore $m_0 = 1$.

**Q3** We know that

$$\begin{aligned} h(x) &= \frac{x^{255}-1}{g(x)} \\ &= (x-1)\prod_{i=5}^{254} x-\alpha^i \\ &= \alpha^{245}+\alpha^{61}x+\alpha^{145}x^2+\alpha^{109}x^3+\alpha^{117}x^4+\dots \\ &+\alpha^{210}x^{246}+\alpha^{147}x^{247}+\alpha^{134}x^2 48+\alpha^{165}x^{249}+\alpha^{76}x^{250}+x^{251} \end{aligned}$$

Where the second equality follows because we know all the roots of both $x^{255}-1$ and $g(x)$. The last equality can be found by putting the first or second equation in a symbolic calculator.

**Q4** Any Reed-Solomon code is maximum distance separable, this means that it reaches the singleton bound. So $d_{H,min} = n-k-1 = 5$.

**Q5** Since a Reed-Solomon code is non-binary, the values it encodes each represent several bits. For example in $GF(2^8)$ every value represents a full byte or 8 bits. This means that if there are multiple errors in the bits of one byte, the code reads it as one wrong value.
On the other hand, if one byte would be entirely wrong, binary BCH would have to deal with 8 errors which comparatively requires a lot of parity symbols.
So a Reed-Solomon code is - generally - better at handling burst errors, which are quite likely in data storage and transmission.

### 3.2 Decoding

**Q1** Since the $C_1$ Reed-Solomon code uses $2t$ parity symbols, it will always be able to correct $t$ symbols. $C_1$ is a $(32, 28)$-code which means that $2t = 4$ and $t = 2$. The code can therefore always correct 2 erroneous bits. Each symbol contains 8 bits. When 2 complete byte errors occur, the code will be able to correct $2 * 8$ bits $= 16$ bits which is the maximum amount of correctable bit errors.

**Q2** The upper bound for the symbol error rate is shown in equation (1). To reach the upper bound, an error pattern consisting of $i$ errors leads to a code word with a fraction $\frac{i+t}{n}$ of the data symbols that are decoded wrong. Here, $p$ represents the probability that a symbol error is introduced in the received code word.

$$\begin{aligned} P_s &\leq \sum_{i=t+1}^{n} \frac{i+t}{n}\binom{n}{i}p^i(1-p)^{n-i} \\ &\leq \sum_{i=3}^{32} \frac{i+2}{32}\binom{32}{i}p^i(1-p)^{32-i} \end{aligned} \tag{1}$$

In equation (2) for the upper bound of the bit error probability, the symbol error probability is recognized preceded by a summation of 2 factors. The first one ($\frac{j}{K}$ or $\frac{j}{8}$) represents the bit error probability of $j$ errors resulting from a symbol error. The second one ($\binom{K}{j}\frac{1}{2^K-1}$ or $\binom{8}{j}\frac{1}{2^8-1}$) is the probability that $j$ bit

errors occur, when a symbol error has appeared.

$$\begin{aligned} P_b &< \sum_{j=1}^{K} \frac{j}{K} \binom{K}{j} \frac{1}{2^K - 1} \sum_{i=t+1}^{n} \frac{i+t}{n} \binom{n}{i} p^i (1-p)^{n-i} \\ &< \sum_{j=1}^{8} \frac{j}{8} \binom{8}{j} \frac{1}{2^8 - 1} \sum_{i=3}^{32} \frac{i+2}{32} \binom{32}{i} p^i (1-p)^{32-i} \end{aligned} \tag{2}$$

When combining these factors and filling in the variables of the $C_1$ Reed-Solomon code, the following relationship between $P_s$ and $P_b$ is found:

$$\begin{aligned} P_b &= \frac{2^{K-1}}{2^K - 1} P_s \\ &= \frac{2^7}{2^8 - 1} P_s \\ &\approx \frac{P_s}{2} \end{aligned} \tag{3}$$

# 4 Audio encoding in CDs

In this section some properties of CIRC (Cross Interleaved Reed-Solomon Coding) are explored. CIRC is the encoding scheme used for CD audio.

## 4.1 Explanation of CIRC components

To maximize its error correcting and detecting capability, CIRC uses several interleaving and encoding/decoding components. Here we give a short overview of all of them and why they are used, in the order they are used when encoding a frame audio. A frame is composed of 12 16-bit samples. Samples of the two audio streams are alternated.

**Delay of two frames** In both streams, even samples are delayed by two frames. This is done to increase the chance of succesful interpolation: errors are likely to occur in bursts, when one or two frames are lost, the lost even samples can be approximated by the odd samples which were encoded two frames further. The same holds for the odd samples.

**Interleaving sequence** To increase the distance between odd and even samples even more, the odd samples are placed at the end of the frame and the odd samples at the start.

$C_2$ **encoder/decoder** After the first interleaving stage, a $(24, 28)$ Reed-Solomon code (inner code) is applied to the 24 bytes. The 4 parity bytes created by doing this are placed in the middle of the frame, between the even and odd samples. This Reed-Solomon code can decode words in which up to 4 symbols have been erased. These erasures are obtained by the outer code communicating the values of these bytes are unreliable.

**Delay lines of unequal length** In the next step, the 28 bytes are distributed over 109 frames: the first byte does not get delayed, the second byte gets delayed by 4 frames, the third byte by 8 frames, and so on. The goal of distributing information this way, is to increase resilience to burst errors. If up to 16 frames are lost, this corresponds to a loss of only 4 bytes in the every frame. As explained above, the $C_2$ decoder can correct erasure of 4 bytes in a frame.

$C_1$ **encoder/decoder** In this stage, a $(28, 32)$-Reed-Solomon code is used to add 4 parity bytes to the frame to obtain a 32 byte frame. These parity bytes are used to correct single symbol errors. If more errors are detected during decoding, this information is passed to the $C_2$ decoder which will ignore the symbols from this frame.

**Delay of** 1 **frame and inversion** Finally, the odd symbols are delayed by one frame to minimize the impact of random errors. The 32 parity bits get inverted. The reason for this inversion is to obtain non-zero parity bits for zero code words. This should improve data readout during silent moments.

## 4.2 Implementation

We implemented the CIRC encoding and decoding scheme in the `AudioCD.m` file accompanying this document.

### 4.3 Maximum correctible burst length

As explained in the $C_2$ decoder paragraph above, the longest burst which will always be succesfully corrected has a length of 16 frames, or $16 * 12 * 16 * 8 = 3096$ bits. As this encodes 96 samples per stream (left and right) and these streams are sampled at $44.1kHz$, this represents $\frac{96}{44100} = 0.218$ seconds of audio. Another way to obtain this figure is to look at the movement speed of the laser. It can be found online this laser moves at $1.3m/s$ which allows it to read 75 blocks every second, every block containing 98 frames. Therefore the time it takes to read erroneous 16 blocks is $\frac{16}{75*98} = 0.218$ seconds.

In $0.218s$ the laser moves $2.83mm$ across the surface of the disk. Therefore, scratches in the radial direction which are less wide than this figure will be able to be corrected.

### 4.4 Interpolation

After decoding the information and deinterleaving the bytes, it is possible some bytes will be wrongly decode *and* known to be wrongly decoded. These bytes can be interpolated using the 8 surrounding bytes. This is possible because the audio was sampled at a rate higher than the Nyquist rate for the frequency band which humans can hear.

The Nyquist rate is twice the bandwidth of the spectrum to reproduce. The spectrum we can hear is $200Hz$ up to $20kHz$, so the minimum sampling rate for exact reproduction is $39.6kHz$. CD audio is sampled at $44.1kHz$. If one ninth of the information of these samples is thrown away, you are effectively sampling at $\frac{8}{9} * 44.1kHz = 39.2kHz$, which is very close to the required rate. Therefore, one could conclude that a ninth byte out of eight known bytes introduces a very minor error.

### 4.5 Performance for different scratch widths and bit error rates

In the following two paragraphs we will test the performance of different configurations of $32, 24$-codes for different types of errors. Configuration 1 is the CIRC configuration as described extensively in the above. Configuration 2 are two concatenated Reed-Solomon codes ($32, 28$ and $28, 24$). Configuration 3 is a single $32, 24$ Reed Solomon code. We will test for scratches of different widths and for bit errors with different bit error rates.

#### 4.5.1 Scratch width

As can be read from Table 1, CIRC encoding performs really well for all tested scratch widths, while the other configurations perform only slightly better than no error correction at all (configuration 0). This is of course a direct result of the interleaving done by CIRC encoding. The Reed Solomon codes used can only correct errors when a few samples per code word are lost. Spreading samples belonging to the same code word over the disk greatly increases scratch resistance.

| | 100 bit scratch | 3000 bit scratch | 10000 bit scratch |
|---|---|---|---|
| Configuration 0 | 0.000194 | 0.00497 | 0.0165 |
| Configuration 1 | 0 | 0 | 0 |
| Configuration 2 | 0.000403 | 0.00432 | 0.0154 |
| Configuration 3 | 0 | 0.00445 | 0.0154 |

Table 1: Chance of undetected errors + chance of failed interpolation for different configurations and scratch widths

### 4.5.2 Bit error rate

As can be seen in figures 4 and 5, the $32, 24$ Reed Solomon code (configuration 3) performs better than the other configurations for any bit error rate. This holds even when taking into account the undetected errors (configurations 1 and 2 will detect all errors). We can conclude that concatenating two RS codes comes at a price. When the bit error rate drops below 1%, the difference between the concatenated codes and the other code starts to decrease rapidly.

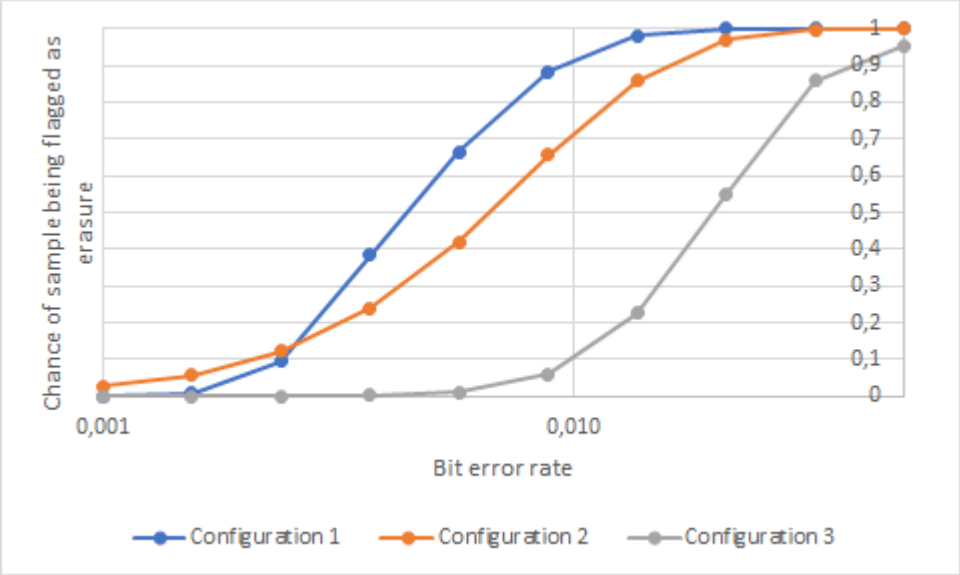


Figure 4: Chance of sample being marked as an erasure for different bit error rates and different configurations

## 4.6 EFM

After the CIRC encoding and before being written on the physical disk, the data goes through a final step, EFM modulation. In this step, every 8-bit symbol is transformed into a 14-bit symbol by using a simple lookup table, hence the name **e**ight to **f**ourteen **m**odulation. The 14-bit symbols are such that every pair of ones is separated by at least two and at most ten zeros. This is done because when writing to the disk, a one is written by changing from a pit to a land or vice versa. (Pits are the etched away parts of the disk, lands are the parts which are not etched away by the laser.) Because of this modulation scheme, pits and lands have a minimum length of three and a maximum length of eleven. This minimum length reduces the demands on the encoding and decoding hardware. If an error is detected in the EFM decoding, erasure flags could get passed to the outer decoder of CIRC.

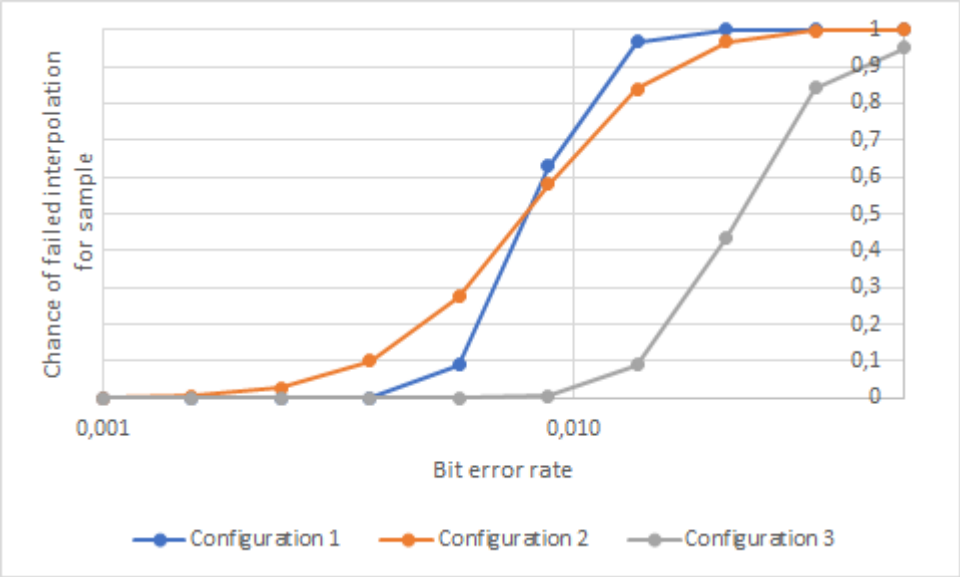


Figure 5: Chance of sample being wrongly interpolated for different bit error rates and different configurations

# 5 Conclusion

The audio CD frame consists of a subcode and CIRC encoded audio samples. The subcode is divided in different channels. The P-channel is used to indicate the start of a track. Using the Q-channel, a CD player is able to know the starting positions of the tracks and the total playing time of the audio CD.

For source encoding, we use a $(32, 28)$ Reed-Solomon code over $GF(2^8)$. By this digital encoding, we can make the data more robust to errors. Furthermore we found that Reed-Solomon codes perform better at burst errors than BCH codes. The $C_1$ Reed-Solomon code is able to correct op to 16 erroneous bits. Also, it appears that the bit error rate equals half of the symbol error rate.

While Reed-Solomon codes are effective at recovering from random bit errors in the code, they do not perform at all when complete code words are lost. This loss of complete code words happens in burst errors, which do occur in practice. When writing audio to a physical disk, scratches are a type of burst error. To mitigate the effect of burst errors, CD audio encoding uses an interleaving scheme, CIRC. In our simulated tests, CIRC is very effective at recovering from scratches, while sacrificing a bit of its random bit error correction capability compared to a code without any interleaving.