

Beyond Bit Perfect: Advancing the Art of CD Reading and Playback

Introduction

Since the advent of digital audio on optical media with the introduction of the CD in 1982, music listeners and designers in the audiophile community have faced continual challenges in attempts to improve the quality of music reproduction. The primary culprit has been identified as “jitter”, which refers to errors in the timing of when the laser sees the sample and begins to read for bits. Also, we use the term “Bit Drift” to describe an error in timing when reading the bits themselves.

In this white paper, we will revisit the common practice of reading digital optical media (with emphasis on 16 bit audio media). We will introduce our process, called Memory Playback, and propose how it will accomplish the following:

- Eliminate error correction and demonstrate a flaw in the original code
- Remove Random Access Memory (RAM and caches) to eliminate odd harmonics created in the DAC
- Use only Solid Memory to play only music bits in their original order
- Preserve inter-relative bit timing that is randomly lost in all optical media
- Raise 16/44 performance to “analog” levels required for High End Audio
- Provide the popular convenience of a stand-alone music server that plays original master quality CD extractions, with nearly unlimited support for Hi-Rez in the future

In the subjective experience of music listening, digital audio was once, and still continues to be, perceived as unnatural to the human ear, due to the replacement of lost data through artificial means. As longer words and faster sampling evolve, this has become less of a problem due to innovations in SACD and DVD audio/video.

Despite the apparent universality of digital audio media worldwide, the High End audio community still favors the LP, due to analog’s perceived superior sonic capability. In fact, audiophile vinyl recordings comprise an LP category unto itself, consisting of re-mastered recordings pressed on extra heavy vinyl.¹

Of inherent value to the history of analog in the High End is the presence of vacuum tubed gear. Tubes enjoyed resurgence in the 1970s, appearing in pre-amps, phono amps and amplifiers, after transistors had “antiquated” them in the mass market. Solid state designs actually boasted of having more transistors than the competition, the antithesis of

¹ .<http://audiophilewiki.org/index/php/wiki/Audiophile>, and Classic Records, <http://classicrecords.com>

"less is more", which refers, in this case, to less circuitry in the signal paths. Less circuitry is a favored practice within the High End.

Tubes first entered digital audio (CD players) with Theta's *California Audio Labs* product, the Cal Tempest I, in 1986.² Tube design was very successful, as digital audio coloration is not unlike transistor audio coloration, and the two, when combined, created a bright, harsh sound. Tubes, sounding warm and sweet (although that is acknowledged as a coloration, too) were thought to be closer to live music. This successfully offset digital audio coloration, resulting in probably the best digital audio sound to date. Many tubed CD players followed.

Audiophiles also have a niche category of recording companies that exploit a variety of media. Audiophile recordings, using unique recording techniques and advanced technologies, target a selective audience of listeners. Companies such as Reference Recordings and Chesky Records, both founded in the late 1970's, have built their reputations on the quality of their recordings.³

Perhaps nowhere has the audiophile's quest for the most faithful reproduction of the live musical experience been more passionately pursued and thoroughly documented than at *The Absolute Sound*. TAS is the High End journal/magazine founded in 1973 by Harry Pearson, currently Chairman of its Editorial Advisory Board.⁴

Many attempts to make improvements to the sound of the CD have developed over the past 25 years. Hardware has trended towards increasing the speed of sampling of the data on the recording, and improvement of the accuracy of clocks. Removable media that use faster sampling rates and deeper bit depth than CD include DVD Audio, first distributed in 2000, Blu-ray, which entered the market in 2003,⁵ and SACD ("Super Audio CD"), Sony's proprietary digital audio format, also launched in 2003. In addition, room treatments and gear "tweaks" to improve overall sonic performance have enjoyed limited popularity.

Behind the issue of distortion in the sound of the CD lies the accepted standard for mass-media error correction, known as Reed-Solomon CIRC (Cross Interleaved Reed-Solomon Coding).⁶ Its universal application from the first production of CDs resulted in the ongoing assumption on the part of the listener that its process yielded no errors at all, in either reading or playback, and was included under the umbrella term "bit perfect".⁷

² *Soundstage Magazine*, July 2000. Also see <http://www.soundstage.com>

³ See Reference Recordings, <http://referencerecordings.com> and Chesky Records, <http://www.chesky.com>

⁴ See *The Absolute Sound*, print version, and <http://theabsolutesound.com>

⁵ See *Blu-ray Disc Association*, wikipedia, http://en.wikipedia.org/wiki/Blu-ray_Disc_Association. The BDA is an industry consortium made up of nine electronics manufacturers, namely Sony, Pioneer, Panasonic, Philips, Thomson, LG Electronics, Hitachi, Sharp and Samsung.

⁶ Vries, L.B, Odaka, K., *CIRC-The Error-Correcting Code for the Compact Disc Digital Audio System*, AES Conference 1st International Conference Digital Audio, June 1982

⁷ See DIYAudio Forums>Top>Source>Digital>What Is Bit Perfect?, May 7, 2005

In 2009, we live in the age of the download, but the lion's share of recorded music can still be found on the CD. Improvement of its performance therefore remains important, and in January 2009, the International Herald Tribune reported:

“Despite the growth of online music sales, CDs remain by far the most popular format, although that hold is slipping. 361 million CDs were sold in 2008, down almost 20 percent from the previous year. About 84 percent of all album purchases were CDs, down from 90 percent the year before.”⁸

Improving the sound of the CD through Memory Playback will be explained within the context of the use of digital technology up to the present day. We will explain why Memory Playback is fundamentally superior to all optical disc reading, as well as computer and portable media downloading.

A Market for the High End Music Server

The trend towards storing and playing music on music servers is well established in the mass market, through the use of compressed, usually downloaded, files like the MP3 or equivalents (WMA or AAC) in portable media players (UPNP) like the iPod, which launched in 2001. As of September 2008, research estimated a total of 173 million Apple iPod units sold worldwide.⁹ The hand-held media player is convenient, and eliminates the need to store and handle discs, since discs require external housing that take up space, are vulnerable to damage, and inhibit portability. There are hard-drive based music servers on the market as well, such as the Olive Opus, introduced in 2005, which utilizes CD ripping, storage and playback of music from a computer's hard drive.¹⁰

According to The Consumer Electronics Association's *Digital America 2008*,

“Eventually music CDs, or any form of music-bearing physical media for that matter, will be as useful as Edison's wax cylinder, some visionaries contend. Eventually, packaged media could be replaced entirely by ‘music servers in the sky’, banks of interactive servers accessed remotely from Internet-connected home stereo systems and PCs to download or stream almost any song ever recorded.”¹¹

But what of the High End? We can expect trends to continue to dominate the consumer electronics market as a whole. However, there are steep challenges presented to the audio industry in its effort to capture and hold the affluent niche market, consisting often of high net worth customers who spend tens of thousands of dollars on gear, in a quest for the highest quality. In a time of shifting economic trends and diminished resources, in order to qualify as High End, strict performance standards need to be clearly defined. Those standards should demand that products go far beyond subjective sonic gains

⁸ Sisario, B., *Album Sales in U.S. Dropped 14% in 2008*, International Herald Tribune, January 1, 2009

⁹ See http://en.wikipedia.org/wiki/Apple_ipod

¹⁰ See olive.us, the website of Olive, Inc., manufacturers of computer-based music servers

¹¹ *Digital America 2008*, The Consumer Electronics Association, page 12

hidden behind expensive cosmetics. Without real technical proof to back up a manufacturer's claim, the consumer risks walking into a gilded trap.

As evidence that the discerning listener has not yet gone the way of the Victrola, in the summer of 2007, the Consumer Electronics Association, surveying a sample of people who listen to digital files on home computers, found that almost 47 percent of the respondents preferred quality of sound over quantity of music.¹²

Audio reviewer Anthony Cordesman, in September, 2008, writing for the UK publication *hi-fi+*, focused on the survival of the High End market, in which he called for a new definition for the High End of the future. Its requirements would cover four main categories, including multichannel recordings, digital audio music servers, surround-sound recording techniques and loudspeaker phase and amplitude correction.¹³

Despite the appeal of the conveniences that currently flood the mass market, the more common practices of playing stored compressed files in large quantities on a PMP device, uncompressed files on a hard drive, or the use of caches or RAM, continue to be strongly associated with relatively inferior quality of music playback. The music server, although acknowledged as important, has been largely neglected by the High End, although there have been some attempts to marry mass market appeal to the more selective listening market.

For example, newer model iPods contain storage space for "lossless" files, enabling listeners (who care to) to download "uncompressed" files. However, since lossless files have predictable data deleted, and later re-inserted at playback, resulting in jitter, this, too, has met with significant disdain by High End consumers.

In addition, "audiophile accessories", designed for use with the iPod, have recently emerged, which include:

- Sound-isolating in-ear Headphones (to replace earbuds)
- Headphone Amps
- iPod Docks
- Headphones for home use
- Vacuum tube preamps for use with an iPod Dock (including the Fatman iTube 425)¹⁴

The goal of creating a stand-alone High End music server is to enable audiophiles to easily store very high-fidelity music electronically, retrieving and playing back selections without the degradation inherent to optical media or altered music files. Because this type of music server enjoys the capability of storing data on memory, it is not forced to correct that data in real time through any of the highly compromised, commonly used

¹² Terauds, J., *High-end audio componentry vs. the iPod*, thestar.com, December 20, 2007

¹³ *hi-fi+*, Issue 60, September, 2008

¹⁴ Bell, D., *Tools for the iPod audiophile*, MP3Insider, <http://news.cnet.com/crave>, March 5, 2009

methods. The elimination of the need for error correction is key to the position outlined in this paper, and to a more analog-like sound.

In an effort to integrate ease of operation into the high fidelity experience, when based on a Windows interface, the server can be operated through a platform with which most users are familiar. The greatest variety of the most powerful hardware on the market is designed for x86 (Windows) support; therefore, Windows is the logical choice for any truly High End component. It can be controlled remotely through the use of a tablet PC, notebook PC, or hand-held device. Whether the music originates from records, tapes, CDs, DVDs, or other media, all the user's stored music can be accessed and controlled from one location.

Since the generally agreed objective of a High End audio system is the "faithful reproduction of a live performance", the challenge for the designer is to eliminate as many obstacles to that ideal as can be humanly discerned. To the audiophile, the sonic criteria of a system can include:

- Dynamics
- Frequency Extension
- Imaging
- Soundstage size
- Timbral Realism, and others.¹⁵

The subject of this paper, Memory Playback, eliminates the primary difference between analog and digital audio, which is error correction, and errors of the playback of bits in the original order that it creates. With error correction bits removed, every bit is a representation of a moment of music. Error correction is the practice of adding bits to conceal mistakes or replace dropped bits at the wrong time or sequence. Many contend it doesn't matter, as the system is perfect, or this known fault, "is inaudible". What this claim actually means, and what "bit perfect" accomplishes, will be explained within these pages.

In the listening experience, Memory Playback almost inevitably invokes a response like "it sounds like records". The reason for this is that Memory Playback is not inherently different from LPs, except for resolution. As stated above, all forms of recorded music, through one format or another, represent moments of music. Regarding the differences among forms of recording media, the distinction is over how much they can resolve. Achieving the most accurate representation is the goal. A more complete and satisfying listening experience can be had through recorded media than previously believed possible, and it can take us closer to the elusive target of the absolute sound.

We will begin to explore this assignment in depth through the dissection of the processes of Reading and Playback, beginning with a focus on what makes the digital listening experience unique.

¹⁵ Soo, C., *One Audiophile's View*, The Stereo Times, Commentary, January 16, 2002

Advanced Audio Reading and Playback in Compact Disc Music Using Read Until Right™, Bit Drift Theory and Digital Data Integrity through Solid Sequential Memory Transmission Lines

PART ONE: READING

What Humans Can Hear: The Problem of Bit Order

In 1982, N.V. Philips and the Sony Corporation released the standard for CD digital audio systems. It was designed to have the ability to detect and correct burst and random errors. The correction code is known as the CIRC, or Cross Interleaved Reed-Solomon Code.¹⁶

Reed-Solomon encoding belongs to the family of burst coding, and was never intended for sub-byte level integrity. Burst coding is defined as “byte level correction”. The most obvious outcome of any byte level error correction is the anomaly, or a series of such, based on certain assumptions about the limitations of human hearing.

The first assumption is that of the actual nature of the output, actual in the sense of time, that is, the temporal integrity of contents of the byte. This paper will explain briefly why Reed-Solomon encoding is blind to bit-level data, and therefore incapable of Bit Integrity (BI), as are all burst codes, the family to which Reed-Solomon encoding belongs. In addition, further assumptions take place based upon the limits of human perception being 15temporally restricted to that which transpires during any given 25us period or less.

It was this decision, in fact, that was responsible for the error corrective devices to create, once decoded to analog, complex and unrelated harmonic content based exclusively upon the use of random, temporal reads in the sub-byte level domain. We contend that it is these in total that are responsible for, and manifest as, the commonly used term for all things abhorrent in digital audio: “digititis”. The problem manifests as a sonic flaw, due to the fact that error correction is blind to the actual location and order of the bits.

Overview of the Uses of Reed-Solomon

Error correction is the replacement of lost data through artificial or “redundant” data targeted at the byte level. The more bits that can be passed through any system, the faster that system is deemed. Reed-Solomon error correction cannot actually correct individual bits, but rather, it can only replace groups of bits known as “bytes”, “frames” or “blocks”; all three terms being interchangeable and referring to the same thing. When Reed-Solomon replaces dropped bits (no data), it replaces a grouping of bits with no real ‘knowledge’ of where the bits were (in time), or what order they were in.

It only replaces the quantity of bits. This is to say that it can replace a byte (a group of 8 bits) without ascertaining any information about the bits. The size of these groups is predetermined by the minimum “resolution” required by the system.

As discussed above, Reed-Solomon codes are burst (byte, not bit) based error-correcting codes. They have a wide range of applications in digital communications and storage. Reed-Solomon codes are used to correct errors in many systems, including:

- Storage devices (including tape, Blu-Ray, CD, DVD, barcodes, etc.)
- Wireless or mobile communications (including cellular telephones, microwave links, etc.)
- Satellite communications
- Digital television
- High-speed modems where bit count takes precedence over Bit Integrity (BI)¹⁷

The wide-ranging RS codes also, by extension, demand different RS code strengths in order to meet a given purpose without using excess space or bandwidth.

The Audio CD Challenge: GER and Probability of Vicinity

The CD structure itself demonstrates the RS code as an odd choice for protection of the data. The reason is that it can protect bytes of bits, but not bits themselves, yet audio resolution is determined in bits.¹⁸ It cannot ascertain any information within the Bit Period of 25 microseconds, yet the designers of the structure determined that human perception is 10X greater than the generally recognized upper limit of hearing at 20,000 Hz or less. A 16X oversampling system would enjoy a 2.5 microsecond bit period and many would say that this is 5X less than can actually be heard.¹⁹

The General Error Rate, or GER (that is, the rate of errors that occur due to a modulation inherent to the system), is a determinant of all time-related errors beneath the Reed-Solomon resolution floor in any given system; in other words, at a rate that is undetectable by the error correction of the system. It is normally defined as errors at a frequency of occurrence in a particular system. A GER should be higher (or lower) than is audible in an audio-based system.

Why? Because if the cyclical errors occur in ultrasonic or infrasonic periods, they are much harder to detect, and are therefore less likely to offend the listener. The Sector Rate of 75 Hz (explained below) is one example of GER occurring at an audible frequency.

Probability of Vicinity, or POV, is a method of predicting the vicinity of a bit in a given, finite field, also known as a Galois Field. It is extremely helpful in predicting Bit Drifts,

¹⁷ Bit Integrity (BI) is a mathematical derivative that represents how many bit drifts, or errors in bit read times, have occurred.

¹⁸ See Appendix A.

¹⁹ See *Behold Electronics* website, <http://behold.eu>

which we define as the anomaly of a bit being read at an instant in time that is NOT the rising edge of the sample oscillator.

GER is important, because unlike other error measurement devices (like Bit Error Rates, or BER, and Block Error Rates, or BLER), GER modulates a composite of these to reveal the errors per sector or a section of data that is fed through the system. In other words, all cyclically occurring data errors are quantified.

As an RS code is a burst code, no information on bit drifts can be ascertained. In audio CDs, the quanta are sectors. Sectors are played at only 75hz. If the .cda data is copied, inverted and imposed upon a 75hz signal differentially, only the phase-shifted data (that is, bit data treated as a modulated 75hz analog signal) will remain. This data is of no inherent value in of itself, but it can help create a predictable environment for sub-byte information and anomalies caused by bit drifts. Once ascertained, we may correct bit timing errors to any degree we wish, post-DAC.

In examining GER, the following points are significant:

- Bit Error Rate tells you how many bits were lost over a given period of time
- Bit Drift (covered in more detail later in this paper) is a mathematical function of a bit being sent to the pipe at the wrong period of time, when the sample period was perfectly detected (zero jitter)
- Block Error Rates tells you how many blocks (bytes) were lost over a given period of time
- Jitter measurement tells you how late the laser was in reading the sample of bits in blocks

GER combines the information of the above (bit, block, and jitter) into a “quality” figure we use to establish an overall level of corrected data in any given system.

What Colors Our Perceptions

Following the adoption by Sony and Philips of the error correction codes in the early 1980s, a documented series of standards, sometimes referred to as the “Rainbow Books”, were formulated for a variety of applications. The standard for CD-DA, or digital audio, known as Red Book, is only one of several. The two most commonly known, Red Book (the “relaxed” standard) and Yellow Book (the higher standard), define how much error we can tolerate for digital audio and CD ROM., respectively.

In addition to Red Book and Yellow Book, Green Book applies to CD Interactive for multimedia, and Orange Book to CD-MO (magnetic optic) as well as CD-WO (write once). There are several others.²⁰

The Red Book standard for CDs is for music CDs only. It allows the error correction bits to occupy only 33% of the CD. The rest is for music data. The Yellow Book error correction is far more accurate and comprehensive, being used for CDs with computer programs on it, and occupying 50% of the CD space.

Red Book CDs are so poorly corrected that if used for computer programs, they would crash a computer. From Recording to CD – More Than Meets the Ear, by Lionel L. Dumond:

“It’s interesting to note that despite the complex error correction methods used on audio CDs, error correction on CD ROMs (data, or the Yellow Book CDs) is even more complex. This is because while an unrecoverable error on an audio CD might at worst result in an annoying skip during playback, a similar error on a CD ROM might corrupt a crucial data file which could render the entire software package useless. Of course the more robust error correction technology required by the CD ROM takes a lot more room to encode, as well.”²¹

Had Yellow Book error correction been used for music CDs, correction would have been nearly perfect, according to today’s definition of minimum human quanta of perception. Yellow Book error correction is 1000X deeper (i.e., stronger) than Red Book.

An interesting parallel effort was the corrective device that predates the “books”, albeit numerically unrelated, which is the audio-compatible **SMPTE** Longitudinal TimeCode (LTC) signal. It uses the bi-phase checking technology approximating the results of Yellow Book depths. Used for motion pictures, it was implemented by the Motion Picture Academy in 1967 and is a classic representation of assumption in a mathematical domain, where truncation would have been the more prudent route taken. SMPTE refers to the Society of Motion Picture and Television Engineers, an internationally recognized standards developing organization, founded in 1916.²²

The modulation scheme, frame-rate and word length determine the range of pulse frequencies required by the LTC signal. Ever mindful that in SMPTE, at normal speed, 960 Hz (zeroes at 24 fps) to 2400 Hz (ones at 30 fps), operates at about 12X the rate of frames on the music CD, it was still considered quite audible to the Motion Picture Academy in 1967.

²⁰ Korst, J., Pronk, V, *Compact disc standards: an introductory overview*, Philips Research Laboratories, Eindhoven, The Netherlands, October 1994, reissued for Multimedia Systems, Springer-Verlag, 2005

²¹ Dumond, L.L., Recording to CD: More than Meets the Ear [online], Newport Beach: Digital Media Online, Inc., 2000

²² See <http://www.smpte.org>, website of the Society of Motion Picture and Television Engineers

SMPTE, ahead of its time, is the digital data standard used in the movie industry that early on recognized that the music CD standard (not yet in existence) was too poor for films, adopting a standard 200% more accurate than the standard later adopted for the music CD.

This serves as evidence of license taken by Philips and Sony in 1982, since Philips and Sony purported an RS code 1000X stronger than the code Philips eventually used.

Once the standard had been established, Philips asked Sony to reduce the strength of their ECC in order to fit the entire Beethoven's 9th Symphony by Furtwangler on a single CD. In this arbitrary decision, in order to add the 9th Symphony, Sony had to subtract ECC resolution.

Today, the High End's accepted method of compensating for the weaker code for CDs is revealed by the presence in the electronics market of 16X oversampling. In 2008-2009, it was popularized (for those who can afford it) by manufacturers including Behold of Germany and Dodson of California (which use 16X oversampling or 16 x 44,100hz).²³

16X oversampling is not all that distant from SMPTE, the standard created in 1967 in Los Angeles. The 16X oversampling machines are often considered to be the finest in the world by many audiophile magazines and e-zines (See *The Stereo Times*, *Stereophile*, and *The Absolute Sound*, 2002-2006), with the sole significant advantage they own over lesser machines purportedly being their 16X oversampling capability. See the Insert *Oversampling*, on page 11.

We will briefly revisit the areas of temporal distortions inherent to the Reed-Solomon burst code error correction.

²³ See Behold Electronics, <http://behold.eu> and Dodson Audio, <http://www.dodsonaudio.com>

OVERSAMPLING

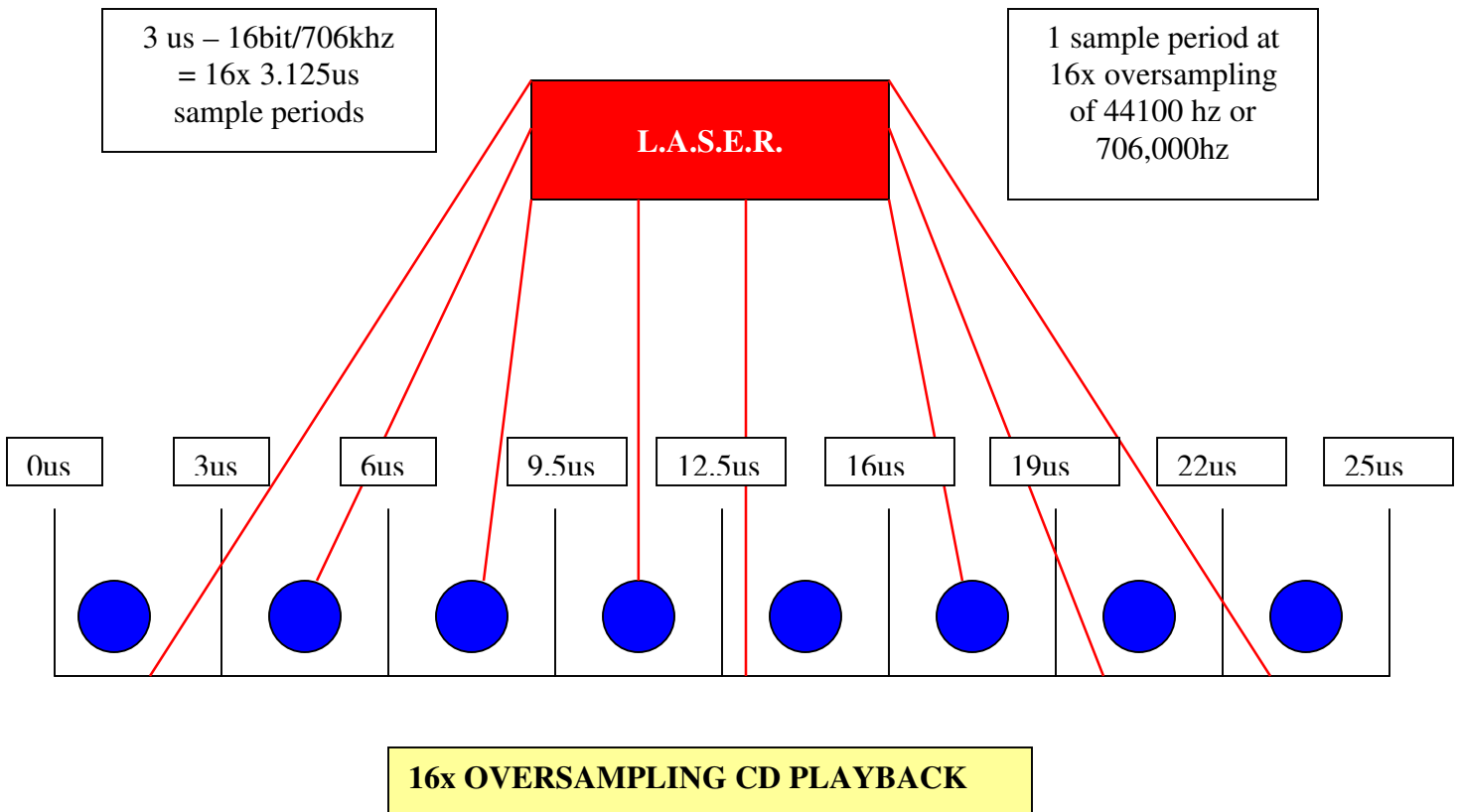
It is commonplace to upsample (or oversample) in order to subdivide the sample period, and limit the time period that a bit can be read, thereby extending the accuracy of the position of a bit in a moment of time, known as the MOE, or “Moment of Event” of a bit’s recognition. See Page 15 for details.

Upsampling as high as 16X, or approximately 705.6 kHz (more commonly 768 kHz or a multiple of 48,000 Hz), will always be restricted to synchronization of some portion of the sample period. Therefore, it would be subject to MOEs as well, albeit reduced errors, by limiting the time available to create a MOE.

From our point of view, the use of ANY burst code recreation technology will have the maximum temporal errors, and should be avoided at all costs. The random phase shifts, once converted, are the stuff of “digititis” itself.

However, upsampling/oversampling is an expensive, blanket approach to ascertain the exact position in time of the bit to be read. As a technology, it would be considered “dumb” (mathematical terms for static vs. dynamic mathematics, not derogatory) as it has no interactive capability. Rather, it merely fragments the period in which errors can take place.

There are much more sophisticated and efficacious alternative solutions, but as these were invented over a four year period from 2003-2007, they are not yet well known to most digital engineers. See below.



The Solution is the Problem: Reed-Solomon Code Abilities and Limitations

How the RS code is determined is most telling of its limitations. What can also be determined is its ability, its inability, and the space it requires in the field.

A Reed-Solomon code (RS) is created by oversampling a polynomial within a Galois Field (finite space). Much speculation has ensued, from various sources within the audio community claiming a degree of technical awareness, over claims that RS codes are “perfect”, or rather, “bit perfect”. However, in mathematics, we determine degrees of acceptable correction ability, and loss. RS codes are no different, and are not perfect. Nothing is. Rather, RS codes enjoy a perfection of efficacy that is predetermined and written into the code, according to need.

They are designated by a nominator indicating the theoretical maximum depth of the resolution and a denominator indicating the actual resolution (in audio optical discs), minus any bits allocated for common parity verification. Parity checking is the only sub-byte level (or “bit level”) error corrective device in the code, however limited it may be, it requires only two bits to effect a 50% solution for a quantifiable correction of a byte.

Reed Solomon codes are a subset of BCH²⁴ codes, and are linear block codes. A Reed-Solomon code is specified as RS(n,k) with s -bit symbols. Please note that “symbol”, “block” and “byte” are used interchangeably throughout this dialog. (See Appendix B for more details.)

To summarize what Reed Solomon accomplishes, a predetermined number of bits are added to the number in the original data, with the amount added determined by the depth of correction required. About 50% of total bits in computer programs are error correction, and about 33% in music data are error correction.

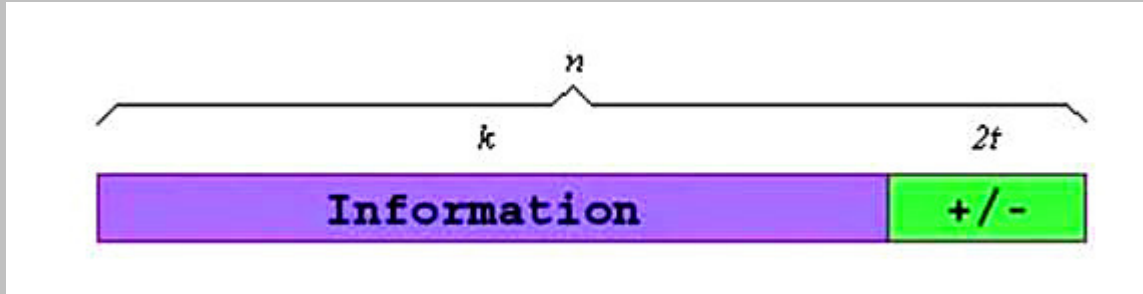
RS codes are encoded on the fly, but decoded from cached data. Many errors are created in an on-the-fly configuration, due to the imprecision of a cheap lens, motor and mechanical limitations. The data is uploaded in fragments to a cache, so that it can be examined for errors and have bad bytes replaced with recreated bytes.

There are several reasons why this is problematic. Caches, like RAM, populate non-sequentially, yet music data requires sequential playback and is extremely sensitive to temporal errors. If caches could be replaced with sequential populating memory, they would run too slowly for on-the-fly correction. Additionally, as error correction is byte-level, it is only sensitive to the bit count in the byte, not the ordering. Expressed mathematically, the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol codeword. There are $n-k$ parity symbols of s bits each. A Reed-Solomon decoder can correct up to t symbols that contain errors in a codeword, where $2t = n-k$.

²⁴ Constellation Networks Corporation, Definitions, <http://constellationnetcorp/definitions.htm>: BCH - A type of block coding of data words named after the inventors Bose, Chaudhuri and Hocquenghem.

THE REED-SOLOMON CODEWORD

The following diagram shows a typical Reed-Solomon codeword (this is known as a Systematic code because the data is left unchanged and the parity symbols are appended):



The popular Reed-Solomon code is RS(255,223) with 8-bit symbols, as computers and CD audio are based upon the 8 bit/byte relationship. Each codeword contains 255 code word bytes, of which 223 bytes are data and 32 bytes are parity. For this code:

$n = 255$, (see below)

$k = 223$, $s = 8$

$2t = 32$, $t = 16$

A decoder can correct any 16 symbol (byte) errors in the code word, i.e., errors in up to 16 bytes anywhere in the codeword can be automatically corrected.

Given a symbol size s , the maximum codeword length (n) for a Reed-Solomon code is $n = 2^s - 1$

For example, the maximum length of a code with 8-bit symbols ($s=8$) is 255 bytes.

Reed-Solomon codes may be shortened by (conceptually) making a number of data symbols zero at the encoder, not transmitting them, and then re-inserting them at the decoder. The (255,223) code described above can be shortened to (200,168). The encoder takes a block of 168 data bytes, (conceptually) adds 55 zero bytes, creates a (255,223) codeword and transmits only the 168 data bytes and 32 parity bytes.

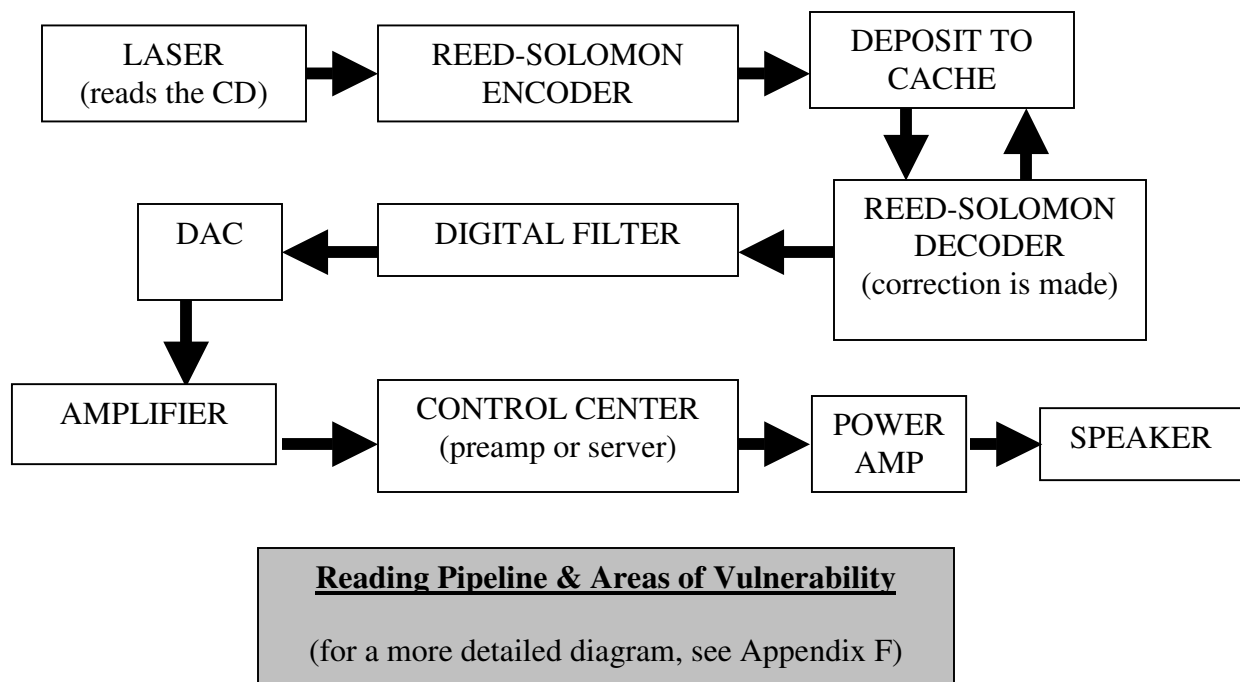
The amount of processing "power" required to encode and decode Reed-Solomon codes is related to the number of parity symbols per codeword. A large value of t means that a large number of errors can be corrected, but requires more computational power than a small value of it.

Error correction is only possible during the time period that the data is uploaded to writable memory RAM. This RAM is called the cache.²⁵

As this data must be changeable (writeable) in order to add correction bits, the time allowed for corrections is directly related to the length of time the cache can store this uploaded data. Larger caches hold more data and longer time periods for correction. After the time period of the cache has expired, the data is fed through, and a new section is uploaded from the CD.²⁶

The predictable criticism of the above is that the speed of RAM and caches is too great for its random rearrangements bits to be heard. This could be said of virtually every improvement of CD audio since the term “bit perfect” was coined.

We contend, rather, that the results of the technologies expounded upon in this document are all audible, but not to the same degree. To that end, nothing shall be assumed beyond human perception here, and the digital engineer may decide to dismiss one and embrace another, according to its value in tradeoffs, or what he or she sees as the desired ultimate sonic result.



²⁵ See <http://en.wikipedia.org/wiki/Cache>, and <http://en.wikipedia.org/wiki/RAM>

²⁶ Reed-Solomon codes may be shortened by (conceptually) making a number of data symbols zero at the encoder, not transmitting them, and then re-inserting them at the decoder. While this is in wide use, it bears no meaning towards the subject of this article. Also see Appendix C.

Errors of Temporal Irregularities

What cannot be corrected by Reed-Solomon codes are generally, but not exclusively, errors in time, and any remunerating of the natural bit order.

Timing errors occur in the post-caching environment and as far down as the DAC itself. In fact, errors in time occur in the creation of the sample itself, as well as the architecture of a given system, and as these are all post-reading errors, they are not detected and/or cannot be corrected (by burst code corrective devices). Perhaps the most heard but least heard about enemy of fixed hardware systems is noise.

The following five topics illustrate likely data distortion and corruption that are unaddressed by Reed-Solomon error correction or Parity correction (or RS255/233 specifically). These are well-known limitations to the science, but are considered to be permissible, inaudible, or impossible to address within the limited size of the audio media.

1. Rise Time of the Sample

Initially, the problem has to do with the rise time of the sample period. The rise time is defined as the time period from the beginning of the clock's sample at zero volts, to the full amplitude of the clock's sample (3-4 volts). More accurately, the two points are the time required to traverse between ten percent and ninety percent of the full rise amplitude.

The time constant is the time it would take for the variable to traverse the full rise amplitude, if its rate of change was the same as at the very start of the rise response. The time constant approximately equals the rise time divided by 2.2. Where t_c is the time constant in milliseconds, the formula for -3 dB cut-off frequency f_c in kHz is: $F_c = 1/2\pi T_c$

In practice, the nominal time is more aptly defined as the root of the sum of the squares of the rise time and the sample time. On the music CD, the frame rate variable is a disaster mathematically. Bit level variants may be indiscernible (insofar as correction is concerned) under the mantra of 'bit perfect' at an effective frequency of only 75hz (see below)!

Consider this: All clocking and synchronization, no matter if it be a cheap slaving from the cyclical redundancy checks of the disc itself, or a \$70,000 atomic clock, are subject to the same errors in TIME that are described above, as the rise time determines when clock sync actually begins.

Fortunately, this bit corruptive event is amplitude-based, and as such, occurs less frequently if the amplitude of the data stream is kept at least 3db above the noise floor. This is predictable according to the formulaic representation described in the Insert *Coding Gain and Noise*, which appears on page 21.

However, CDs of questionable quality or those being played in a noisy environment (more with CD-Rs) will be subject to Rise Time Jitter at the rate of the Time Constant described above.

A mathematical, temporal relationship between bits in the byte was developed to help describe their random positions in space and either correct the results once decoded or stream it to sequential memory and correct it at the source: the MOE.

2. Failure is the Only Option: ECC and Distortions in Time

The first failure of burst (Reed-Solomon) error protection is best illustrated by examining The Bit-Period, the time period in which a bit may be read. It is mathematically defined by the relationship of three points in defined space. This temporal relationship, or the definitions derived from examining the beginning of the sample period, the periods a bit may be read, the time a bit is read and the time at the end of the sample, is defined by a mathematical composite known as the MOE (Moments Of Events). (Please see the chart and notes labeled *An Illustration of Time Errors*, on page 16.)

What we must now address are areas of temporal bit integrity corruption that are, in a very real sense, hierarchical in their import for the music CD. This mandates a scrutiny of the CD structure.

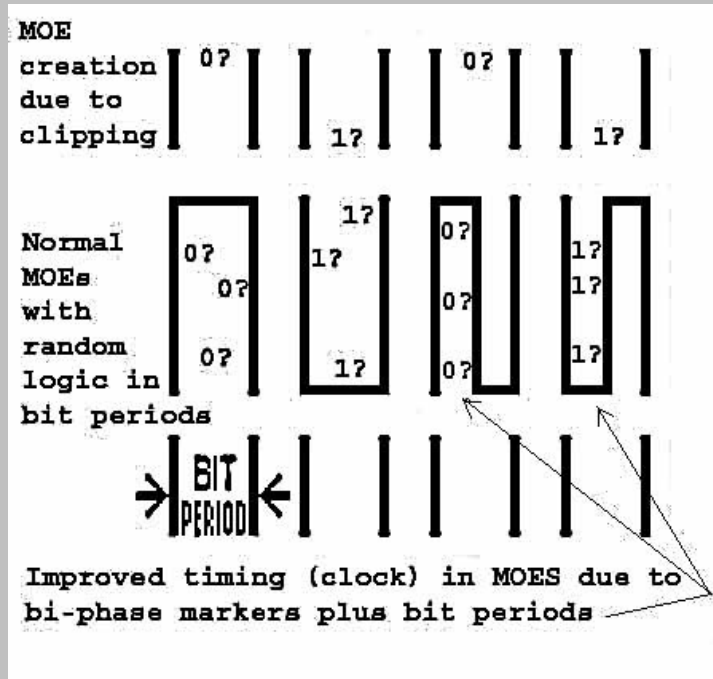
The binary value 0 is represented by a single transition at the start (or 'boundary') of the bit-period (or bit 'cell'). The binary value 1 is represented by a two transitions; one at the start and the second in the middle of the bit period. This scheme is called 'Bi-Phase Mark' and closely resembles frequency modulation ('FM'); that is, a stream of binary ones is represented by a burst of audio at double the frequency of a burst of audio, which is used to represent a stream of binary zeros. (Also see Appendix A.)

3. Introduction to Bit Drift Theory

Even though bit perfect would be better termed “bit count perfect” and “jitter” termed “temporal sample error”, a far more deleterious issue is that of “Bit Drift”. These are some of several important audio corrective re-definitions through which an important distinction needs to be made: In current technology, a claim of bit perfect assures that bit counts are correct; however, Bit Drifts within the temporal periods of a sample are most often dismissed as inaudible and ignored. However, in order to make any improvement in 16/44, we must consider every fault, inadvertent or included in the ECC itself, wrong.

We define “Bit Drift” as a distortion of the moment in time between the rise and decay of the sample period in which a bit is successfully read. A bit should be read at the exact center of the bit period. Mechanical systems are unreliable at speeds such as these. The pickup lens has too much diffraction to guarantee a perfect read at the perfect time. The CD itself is 10 cents worth of plastic and aluminum (usually). The error corrective devices cannot read bits, only bytes (the ‘capsule’ of 8 bit groups by which data is sent).

AN ILLUSTRATION OF TIME ERRORS



The above is a simple illustration of several important areas of errors in TIME that are not considered within what burst code protection would consider “bit perfect”. It is NOT. It has severe error potential when no error correction is deemed required.

As shown, the first error and the least, is the rise time of the sample oscillator (the clock). The reason this is important is that if the amplitude is not above the noise floor, and is below any point that could be considered clipping (as clipping can invert logic), it is a small but important area of error.

The clock, regardless of its accuracy, usually synchronizes to the rising edge of the sample period. During that period, the laser is in seek mode, and a return code during any point of the Bit Period will read a 1, and therefore, be considered the initiation of a frame to which the “clock” must synchronize. No return, or a second return of a code within the Bit Period, will read a zero. In either event, we now need to examine the moment of event, or MOE.

MOEs help us predict an inherent error of a temporal nature that is represented in microseconds that are in the inverse proportion to the numeric value chosen to represent the Bit Periods Time.

In CD, this Bit Period is 25us and therefore the MOE potential is during that time envelope. Unlike the SMPTE described previously, it enjoys no Bi-Phase Mark and therefore the phase errors are unmitigated and require more computation; the maximum potential equals the event, so a MOE = 25us or $1/44100^{\text{th}}$ of a second of random, phase error.

4. Symbol Errors or Mis-Bits in the Byte

The description below explains the limitation imposed on CD error correction. The problem is all CD error correction is byte level, not bit level. A byte contains eight bits. For error correction to have worked in the CD, it would have required it have a byte only one bit wide, not eight bits wide. We have no idea what the eight bits were, when they were played, what order they were in, were they 1s or 0s; only that there were eight of them. To that end, bits are almost always read at the wrong time. These temporal errors are often quite small, but in almost every byte, bits are predictably out of time.

We believe this is paramount to the artificial nature of digital audio sound. All recording media differ in resolution, losing some of the original information. But what remains is still original information. Only digital audio actually changes the original information itself, randomly, and without any ability to correct or delete it.

DESCRIPTION OF BYTE CORRECTION vs. BIT CORRECTION

For practical uses of Reed-Solomon codes, it is common to use a finite field F with 2^m elements. In this case, each symbol can be represented as an m -bit value. The sender sends the data points as encoded blocks, and the number of symbols in the encoded block is $n = 2^m - 1$.

Thus a Reed-Solomon code operating on 8-bit symbols has $n = 2^8 - 1 = 255$ symbols per block. (This is a very popular value because of the prevalence of byte-oriented computer systems.) The number k , with $k < n$, of *data* symbols is a design parameter. A commonly used code encodes $k = 223$ eight-bit data symbols plus 32 eight-bit parity symbols in an $n = 255$ -symbol block; this is denoted as a $(n,k) = (255,223)$ code, and is capable of correcting up to 16 symbol errors per block.

The choice of using parity at all proved prophetically poor for recorded music. A bit code of RS255/254 could have resolution to a single bit and recreate bits in literal in space and time. But it would be huge and leave only 33% of the CD for actual program material. With shortening, (see Appendix C), requiring only that all bits be present and in the correct order, and with time/space bit relationships irrelevant, RS255/239 proved perfect for computer programs.

The set $\{x_1, \dots, x_n\}$ of non-zero elements of a finite field can be written as $\{1, \alpha, \alpha^2, \dots, \alpha^{n-1}\}$, where α is a primitive n th root of unity. It is customary to encode the values of a Reed-Solomon code in this order. Since $\alpha^n = 1$, and since for every polynomial $p(x)$, the function $p(\alpha x)$ is also a polynomial of the same degree, it then follows that a Reed-Solomon code is cyclic.

The error-correcting ability of any Reed-Solomon code is determined by $n - k$, the measure of redundancy in the block. If the locations of the errant symbols are not known in advance, then a Reed-Solomon code can correct up to $(n - k) / 2$ erroneous symbols, i.e., it can correct half as many errors as there are redundant symbols added to the block.

The immediately obvious fatal issue is illustrated here, and it is the very crux of the matter of the superiority of memory players vs. mechanical CD players. It is that RS255/239, the 100% corrective code (for bit counts and bit ordering), requires 50% of the CD area to accomplish this efficacy. RS255/223 is allotted only 33% of the CD area. The bit errors can be as great as 1000X worse as a result. That is, RS255/233 can enjoy 100% byte error correction, but has literally ZERO bit error correction.

This was deemed inaudible within the time period of 25 microseconds. That is, if you can hear more than four moments of musical information at 10,000hz, RS255/223 fails.

One symbol error occurs when 1 bit in a symbol is wrong or when all the bits in a symbol are wrong.

Example: RS (255,223) can correct 16 symbol errors. In the worst case, 16 bit errors may occur, each in a separate symbol (byte), so that the decoder corrects 16 bit errors. In the best case, 16 complete byte errors occur so that the decoder corrects 16 x 8 bit errors. Reed-Solomon codes are particularly well suited to correcting burst errors (where a series of bits in the codeword are received in error).

By computer program standards (which use RS255/239, not RS255/223) this is a disaster. It is predicated on an assumption that human perception is really quite poor, and incapable of discerning 25 microsecond delays, and only rise times 4X longer than that.

While this may seem subjective, one need only to take a look at a 5000 Hz square wave of a test CD, if you are certain of Philips' wisdom in 1982. The arithmetic tells only part of the story. It demonstrates a subjective decision, made of allowable errors, inherent to the RS255/223 corrective ceiling. This assumes that no other areas of vulnerabilities or variabilities exist, which, of course, they do. Still more errors are derived from noise occurring while data awaits error correction, and from error correction itself. In both cases, the errors occur in the realm of TIME.

Although inter-temporal bit relationships will not be preserved by Reed-Solomon algebraic decoding procedures, they can correct s -errors and r -erasures. An erasure occurs when the position of an erred symbol is known. A decoder can correct up to t errors or up to $2t$ erasures. Erasure information can often be supplied by the demodulator in a digital communication system, in other words, the demodulator "flags" received symbols that are likely to contain errors.

When a codeword is decoded, there are three possible outcomes:

Scenario 1: If $2s + r < 2t$ (s errors, r erasures), then the original transmitted code word will always be quantifiably and sequentially recovered,

OR

Scenario 2: The decoder will detect that it cannot recover the original code word and indicate this inherent failure,

OR

Scenario 3: The decoder will mis-decode and recover an incorrect code word without any indication.

The probability of each of the three possibilities depends on the particular Reed-Solomon code and on the number and distribution of errors. In order to present a lucid example of where these errors can occur, a brief explanation of how the Reed-Solomon code in question, RS255/223, the code used on the MUSIC CD and RS255/239, the code employed on the SOFTWARE CD, were determined. The RS255/223 has a 1000X greater probability of bit errors than RS255/239.

In the possible outcomes described above, the numbers reveal an environment for bit-level corruption. **Scenario 1** is common; however, it still suffers from the 25 microsecond delays in bit reading described above.

Scenario 2 is less common, and occurs when a CD is damaged or dirty. Interpolation results and although not desirable, it is impressive considering the gaping holes of silence that would be the alternative! This is the only disadvantage of memory players, as they would yield silence in such a state. A system utilizing genuine Memory Playback must employ intelligent rereading to circumvent large drops. In our experience, rereading must be over 100X to enjoy very high bit time/space efficacy. Popular freeware CD rippers like EAC can reread only 20X, due to the inability to alter or augment third party codes.

Scenario 3 occurs in nearly every byte to some extent. As RS codes and measurements for BERs and BLERs cannot ascertain any sub-byte information, only MOE data can provide information on bit timing distortion due to Bit Drift. Bit Drift can also be measured through controlled, parallel DAC measurements of analog phase shift data. In our experience, random 23 degree phase shifts at 4000hz are commonplace.

5. Noise and Hardware

The most heard (but least heard about) bit corruptive element in *hardware-based memory players* systems is noise. A wealth of discreet chips designed for new applications will have more noise and jitter vulnerability (from local noise sources) as they contain additional, internal current sourced buffers for compatibility with a wider variety of ancillary chips and conventional circuitry. As a result, it is usually more prudent to seek out motherboard configurations with dedicated chips directly coupled to subsequent, dedicated chips and so forth, down the pipe. Motherboard chip relationships have far fewer buffers in the paths and pipes, and are designed to use less circuitry, as they are optimized for one another.

Although discreet components vulnerability to local noise is much worse than with motherboards, bit inversion, corruptions, drops and quanta-mechanical disruptions of natural measuring procedures can still occur in both configurations as a result of noise.

The precision with which noise interference is negated in dedicated CPU/MOBO topologies reduces noise-based bit errors 100-fold or more. A brief revisiting of Gaussian Noise distribution and Probability of Vicinity could help make this dilemma clear.²⁷

The beginning of resolving any temporal errors in a given bitstream pipeline system is to first purge it of mechanical devices as much as possible, as they are inherently unreliable.

As the RS255/223 is only capable of correcting large byte errors, it finds its greatest value in hardware-based systems where the errors tend to be near catastrophic; for example, buffer under-runs, write caching, synchronous transfers, conventional tagged queuing with no FUA bit, and of course, physical damage to the optical media itself.²⁸ Those would include discreet FPGA topologies using any form of Random Access Memory. Most motherboards enjoy dedicated Spectrum Spreading as a formidable deterrent to noise-created nodes and the resultant intermodulation distortions.

However, in faster bit-oriented environments, where bit timing and order are considered imperatives, mechanical and hardware systems are too slow and error prone to be of value. Unfortunately for us, CD audio is such a system, and as such, should only be played or read in a software system, as it has no mechanical liabilities, and it is nearly unlimited in speed. Time and original orders of bits can be achieved over 90% of the time in software systems, and as they are infinitely writable, software systems are millions of times faster than hardware and can scale upwards with little if any hardware changes.

To “qualify” as a true software system, the device cannot cache, or the liabilities of the RAM come back into play. Such a system is not a true memory player, and is greatly exposed to noise environments. (Also see Insert below, *Coding Gain and Noise*.)

SUMMARY of 1 through 5:

The initiation point at which a laser seeks a bit is at the rise time; a fast initiation point that starts the sample period. The time of this point itself is not “instantaneous” (nothing is), and so both the time required for the initiation of the sample, and the level (voltage) at which a clear initiation begins, must be defined. It is preferable that this point be far above the noise floor, which is different in every system, making a nominal choice complicated.

²⁷ See Ziemer, R.E, Peterson, R.L. and Borth, D.E., Introduction to Spread Spectrum Communications, paperback, Prentice Hall, 1995

²⁸ FUA is Forced Unit Access bit, a command to actually bypass the caches and RAM to the source so that there is a degree of intelligence about the source in the event of questionable data. See Pearson, Jonathan, supervised by Kai Shen, *Journaling Filesystem Consistency on Disks Without Tagged Command Queuing*, University of Rochester, 2007

All clocking, regardless of quality, synchronizes itself to this initiation time. Remarkably, what happens to bits in the byte during the sample period is not affected by the accuracy of the clock, as all known information is about the alignment of the clock to the sample, not the bits in the sample.

This “Bit Drift” is the primary cause of errors and “digital sound” or “digititis” in normal CD audio systems that cannot be addressed with error correction. Only memory players can correct these “post-caching errors”, because all of the data is on the memory, and fully writable (correctable). This enables comparative examination, rewriting over errors and enough time to add reread data to achieve a master mirror.

CODING GAIN AND NOISE

Coding Gain explains the performance of error correction codes within an environment of noise. All local environments have noise, and the substrates themselves have an infinite variety of thermal noise. Explaining how these codes behave, misbehave and can be improved upon in an environment of noise is important, as noise is omnipresent and random and as such, resembles a hash code crack. The theoretical works presented here attempt to demonstrate the detriment noise imposes upon digital signals and the various methods in use to prevent corruption.

Reed-Solomon codes preserving data in a noisy environment is their primary advantage. The ratio of a given code’s efficacy is inversely proportional to the noise in decibels of the environment. This ratio is often described as coding gain.

*Example: A digital communication system is designed to operate at a Bit Error Ratio (BER) of 10^{-9} , i.e., no more than 1 in 10^9 bits are received in error. This can be achieved by boosting the power of the transmitter **or** by adding Reed-Solomon FEC (or another type of Forward Error Correction). Reed-Solomon allows the system to achieve this target BER with a lower transmitter output power. The power “saving” given by Reed-Solomon (in decibels) is the coding gain.*

Finite Field Arithmetic is the basis of describing a three-dimensional space. Introducing the element of time creates four-dimensional space, or the behavior of a code in a known space at a known time. This is the crux of new error correction work as bits are in flux, so when a bit was and where it was is how provides enough information to recreate it in the original time frame. That is, the element of time must describe the bit’s position in four-dimensional space or massive phase shifts exiting the DAC may occur at any time.

The Realities of Hardware vs. Software

The primary objective of the dialogues below is an effort to clarify the specific categories of which topologies are in use in a given systemic, and why. They are all legitimate, and as all good design requires, they employ tradeoffs to target a price point.

The oldest form of digital audio players, and even the earliest memory players, relied heavily on hardware. The only true difficulty in this is that hardware players are literally outdated every six months to a year.

Re-reading players require software to be rewritten to scale upwards to exact more and more precise control of the laser. In a hardware player, the ability to improve communication between the laser and the subsequent processing is outdated quickly, as it is fixed.

For example, a few hardware 'memory players' boasted using the popular EAC CD ripper. This ripper has employed a well-known freeware rereading code. However, this code is limited to 20X rereading regardless of whatever it is programmed to do in EAC. (Nova Physics is writing a 300X rereading software for release in late 2009.)

The reason for this is that the communication between the laser guidance and the software is still using RS codes. (Anecdotally, most, if not all, 'memory players' are manufactured in Eastern Europe, but marketed in the United States.)

Software "builds" for computer-based memory players can control the movement of the laser in increments that no hardware can approach, since software can be improved by continually refining the mathematical relationships for laser guidance. Hardware machines have a fixed setting or settings for this, and cannot be improved without literally replacing the hardware itself. It is an outdated approach to the refinement of laser guidance, with literally no provision for bit order preservation.

As such, in many instances, there is lack of understanding of these new technologies. To the contrary, if EAC or a derivative of it is employed, so, then, is ECC.

Hardware: FPGAs and other Outdated Discreet Hardware Implementations

A number of commercial hardware implementations exist. Many existing systems use "off-the-shelf" integrated circuits that encode and decode Reed-Solomon codes. These ICs tend to support a certain amount of programmability (for example, RS (255,k) where $t = 1$ to 16 symbols).

A recent trend is towards VHDL or Verilog designs (logic cores or intellectual property cores). These have a number of advantages over standard ICs. A logic core can be integrated with other VHDL or Verilog components and synthesized to an FPGA (Field Programmable Gate Array) or ASIC (Application Specific Integrated Circuit) – this

enables so-called "System on Chip" designs where multiple modules can be combined in a single IC.

FPGAs are the most widely used and the least desirable/least expensive. Depending on production volumes, logic cores can often give significantly lower system costs than "standard" ICs.

Software: Faster Than Any Hardware - The New Frontier for CD

Until recently, software implementations in "real-time" required too much computational power for all but the simplest of Reed-Solomon codes (i.e., codes with small values of t).

The major difficulty in implementing Reed-Solomon codes in software is that general purpose processors do not support Galois Field arithmetic operations. For example, to implement a Galois Field multiply in software requires a test for 0, two log table look-ups, modulo add and anti-log table look-up. The data rates are for decoding only, as encoding is considerably faster, requiring less computation.

This all changed with the advent of atomic rereading during the development of RUR™, as active frame alignment and dynamic sector overlapping became possible at 70X the nominal RPM of the Compact Disc. It is now (as of 2009) possible to reread 150X in less time than traditional extraction algorithms using Reed-Solomon and ECC (like EAC) can achieve.

Hardware-based topologies are never desirable, as they are slow and read-only, or nearly so. Software-based topologies can be updated almost indefinitely with very little cost for the user, and are generally limited only by the bit-resolution (depth or word length). At this time, 32-bit is the highest resolution available, and is commonly used in professional recording studios. (The Memory Player uses 32 bit architecture as well.)

What Is or Is Not A Memory Player

We have observed that write ordering may not be guaranteed in the sub-byte realm. This is exaggerated by the use of Reed-Solomon error correction and Random Access Memory.

The circumvention of bit reordering is the very crux of what makes any playback system a memory player or not. The issue is that byte levels are 8 bit capsules, which are cached, and the bit count and odd/even orders are then verified. If incorrect, RS replaces the byte with its generic ordering of 1-8, sequentially. If odd numbering is detected, Parity adds a bit to bring it to even. Both replace the bits after the original order was written, or replace the original order altogether. Only bit count is preserved and heard. Bit ordering is not.

To that end, if bit reordering is not absolutely prevented, it is not a memory player. Only Solid Sequential Memory (also known as "Solid Memory") can be programmed to not reorder.

Modern magnetic disks and optical data systems have caches. A cache, being a small block of memory that helps speed operations by buffering data streams to circumvent a few seconds of under-run, allows the disk to respond to read requests without repositioning the head, and to respond to write requests before actually writing the data to the other receiver.

This, of course, is implicitly dependent upon a contiguous journal being written in advance. The disk may even go further than just holding onto unwritten data until the head is in place. It may even change the order of the write operations themselves.

Why? If the writer is going to pass near one block of 0s in seek mode to another part of the disk/RAM, it is in an opportune place to write the 0 data, so it 'empties' its 0's prior to writing its 1's, as this is the nature of random access. With the inclusion of disk caches, the act of writing the journal before the real data arrives becomes a much more complex issue. Should no form of data stream irregularity occur, bits will drop, with no method of recovery open to the system. (Please refer back to FUA, page 21, footnote 28.)

The solution? Never, ever cache.

From the results of the analysis which the above simply touches on, it may be derived the following tenets of prohibitions to obtain maximum fidelity in memory players:

- **NEVER, EVER USE CACHES OR CACHING**

Caching may never be used, as it invokes Gaussian Noise, and fragmentation errors take place (from partial uploading of data), and even bit inversion. In addition, Spectrum Spreading, the most effective technology against noise-based data corruption, is impossible except in CPU/MOBO topologies.

- **NEVER USE RAM AS MEMORY FOR FIDELITY FUNCTIONS**

RAM cannot be used, as it cannot populate sequentially (hence Random Access Memory) and will introduce bit reordering, by nature.

- **NEVER DEPEND ON HARDWARE FOR FIDELITY FUNCTIONS. SOFTWARE IS ADAPTIVE AND EXTREMELY FAST; HARDWARE IS SLOW, OUTDATES QUICKLY AND CANNOT BE UPDATED (IS NOT SCALABLE)**

As they become outdated quickly, hardware suffers from very limited scalability (upgrading existing chips and hardware without replacement).

Hardware-based players are extremely vulnerable to local noise, and cannot implement the fastest preventive measures.

To summarize, caching is amongst the most serious vulnerabilities in digital systems. Its problems include:

- Fragmentation of the data
- Holding data within the local noise fields (other circuitry, clocks, oscillators, etc.), and
- Limiting the time the data may be corrected in to the size of the chance itself, instead of depositing the data as a whole, as is done in true memory players.

Lossless Losses – There’s No Free Lunch

It is also worth mentioning the difficulty of compression of "lossless" files.

An analogy: If a sentence was transmitted with very limited space, you could remove say, every "e" knowing that the receiver will replace the e's on the other side. For example,

"These Melos phono preamplifiers are the finest I've ever heard"

could then become

"Th s M los phono pr amplifi rs ar th fin st I'v v r h ard".

and for transmission

"Ths Mlos phono pramplifirs ar th fi st I'vvr hard".

These temporal reconstructions have the same quantity of "e"s, but the time the "e"s were replaced (after the byte was written to a cache), is incorrect, and will be decoded in the DAC as random phase shifts. By fragmenting the data after it was read, timing for insertion of the "e"s (compiling) is now far more critical, as the chances of jitter have greatly increased and cannot be corrected.

To that end, we consider no compression acceptable.

INTRODUCTION TO PART TWO: PLAYBACK

The Answer To Bit Imperfection: Read Until Right, or “RUR”

A “smart” technology designed as an alternative to oversampling (upsampling) was originated in 2004 by Nova Physics and was coined “RUR™” or “Read Until Right™”. RUR works by exploiting the sector seek period of $1/75^{\text{th}}$ of a second, dynamically.

RUR sends for a return when a frame is detected. Once detected, a mirror of a CD is written to Solid Sequential Memory (Solid Memory™), and the “perfect”, virtual CD structure is created on memory, but consists of all 0s, at this point.

Upon a return code from the frame, seek is initiated and the first sector is available. If there is no return, a 0 is written to Sector 1 and a repeat command is created and executed. In most cases, the static repeat will succeed in a read, as a failure is most often caused by the optical disc’s cyclical motion irregularities/vibrations, or has obstructions that fall off or are circumvented by minute changes in the physical position.

If RUR receives two static repeats, a dynamic reread effort is created. Each sector failure then creates negative and/or positive movement of the laser. The seeks now have the ability to start a sector read in twenty-six different positions and again, send a repeat as a result of no return of the initiation of the sector.

This can result in a DAE as slow as 1X; however, it almost inevitably results in a successful read.

Alternative rereading is effected as a last resort of sorts, until RUR executes a zero bit and the read, abandoned. In order to reduce the potential of IM at 75 Hz, RUR operates at multiples of 75 Hz ONLY.

The final efforts after 148X rereading have been made are to execute sector parallelism readings. In this mode, the three previous sectors are compared with the three subsequent sectors after a failed read. If they match, the missing code can be determined and written. This completes a 150X rereading cycle.

In practice, 90% - 98% readings take place, and occasionally (on very clean CDs) 100% re-readings take place, in which case, a literal recreation of the original master is on the memory.

As described above, all error correction inserts bits into the playback stream out of the original time and out of original order. Only the bit count is correct, not the bit timing or order. This results in bizarre harmonic structures, and irresolvable ambient cues (or no ambient cues at all) that generally separate digital audio from analog. To that end, the primary goal is to remove all error correction bits and capture all dropped bits through rereading.

To summarize the advantages of RUR™:

- Blind (“dumb”) rereading only rereads the same area in the same way, and yields poor efficacy. RUR™ is considered a “smart” code, although it might be better described as an “active” code, whereas blind rereading is “generic CD ripping”.
- RUR™ can reread missed bit areas, and adjust the laser backwards and forwards, overlap entire sections of bad areas, and even adjust laser speeds in an effort to capture missed bits.
- In addition, it reads in multiples of the CD’s sector (frequency of the batches of data) frequency (75hz) such that it nullifies the distortion in the bass (IM distortion) and inevitably resolves greater bass resolution than ECC-based systems can approach.
- Its “efficacy” speeds up rereading and allows more and more rereading in less time. At present, genuine Memory Playback is capable of 150X rereading, and as of this writing (early 2009), 300X rereading is on the horizon. This will be, by far, the highest amount of rereading of any memory player in the world.

Again, RUR™ displays the power of software-based systems, as no hardware need be changed, or is at risk of becoming outdated in an effort to update to 300X rereading, in older memory players.

RUR’S Place In Memory Playback

We propose that the misrepresentation of what is called ‘bit perfect’ has an answer in a three-part process that represents reading and playback state-of-the-art in stand-alone High End music servers. This process consists of:

- Read Until Right™
- Bit Alignment
- Playback through Solid Sequential Memory (Solid Memory™)

The entire process is known as Memory Playback. The Playback portion of the process, with more about Bit Drift Theory, as well as an overview of Solid Sequential Memory, will be covered in Part Two.

END OF PART ONE: READING

Bibliography: Part One

In this paper we have deliberately avoided discussing the theory and implementation of Reed-Solomon codes in detail. **For more detail on Reed-Solomon, please see the following:**

Reed-Solomon ECC

Vries, L.B, Odaka, K., CIRC-The Error-Correcting Code for the Compact Disc Digital Audio System, AES Conference 1st International Conference Digital Audio, June 1982

Wicker, S.B., Error Control Systems for Digital Communication and Storage, Prentice-Hall, 1995

Lin S., Costello, D., Error Control Coding: Fundamentals and Applications, Prentice-Hall, 1983

Clark, G.C., Cain, J.B., Error Correction Coding for Digital Communications, Plenum, 1988

Wilson, S., Digital Modulation and Coding, Prentice-Hall, 1996

Riley, M.J., Richardson., I.E.C., Digital Video Communications, Artech House, 1997

Dunn, C., Hawksford, M.O., *Bits Is Bits*, reprinted by permission of the AES in *Stereophile*, Vol. 19, No. 3, March 1996 (based on the paper entitled *Is the AES/EBU S/PDIF digital audio interface flawed?*, presented at the 93rd Audio Engineering Society Convention, October 1992, in San Francisco).

Also please see the following references:

Digital Audio

Korst, J., Pronk, V, *Compact Disc Standards: An Introductory Overview*, Philips Research Laboratories, Eindhoven, The Netherlands, October 1994

Atkinson, J., *DVD-A vs. SACD*, *Stereophile*, As We See It, November 1997

Walsh, C., *Sony Plant Marks 20th Year With Launch of SACD Line*, *Billboard*, May 17, 2003

Richter, M., *What's This DAE Stuff Anyway?*, (explanation of Digital Audio Extraction), <http://www.mrichter.com/cdr/primer/dae.htm>

Pearson, J., (supervised by K. Shen), *Journaling Filesystem Consistency on Disks Without Tagged Command Queuing* (Masters Thesis), University of Rochester, 2007

Digital America 2008, an annual publication of The Consumer Electronics Association

Pohlmann, K.C., *The Compact Disc Handbook-The Computer music and digital audio series*, 2nd Ed. Madison, WI, A-R Editions, Inc. 1992

Watkinson, J.R., *The Art of Digital Audio*, 2nd Ed., Boston, MA, Focal Press, 1994

Miscellaneous Audiophilia

Audiophile Wikipedia, <http://audiophilewiki.org/index/php/wiki/Audiophile>

Deutsch, R., *Are You A Sharpener or a Leveler?*, *Stereophile*, As We See It, Vol. 32, No. 2, February 2009

Harley, R., *CD: Jitter, Errors and Magic*, *Stereophile*, Vol. 3, No. 5, May 1990.

Guttenberg, S., *Grammy Winning Record Producer Says CD Quality Isn't Good Enough*, (interview with T. Bone Burnett), *The Audiophiliac* (A high end audio blog from Steve Guttenberg), June 11, 2008, <http://news.cnet.com/8301-13645-3-39964576-47.html>

The Absolute Sound, Forums, High End Audio Industry, <http://avguide.com/forums/consumer-electronics-industry/high-end> audio industry

Stereophile, Reference, <http://www.stereophile.com/reference/index11.html>

Positive Feedback, print version, and <http://www.positive-feedback.com>

The Audiophile Voice (High End industry magazine)

Hi-Fi News, and <http://www.hifinews.co.uk>

hi-fi+, and <http://www.hifiplus.com>

Also see the following website and e-zines:

The Society of Motion Picture and Television Engineers website, <http://www.smpte.org>

The Stereo Times, <http://stereotimes.com>

Enjoy The Music (High End Audiophile Equipment and Music Reviews), <http://enjoythemusic.com>

6moons, <http://www.6moons.com>

Stereo Mojo (Information and reviews for the value-conscious music lover), <http://www.stereomojo.com>

END OF BIBLIOGRAPHY: PART ONE

APPENDIX A

CD Structure

CD-ROM Sector, Frame, and Audio Data Information

Advertised CD length (minutes) 74 80

| | | |
|----------------|----|----|
| Sectors/second | 75 | 75 |
| Frames/sector | 98 | 98 |

| | | |
|--------------------|---------|---------|
| Number of sectors | 333,000 | 360,000 |
| Sector length (mm) | 17.33 | 17.33 |
| Byte length (um) | 5.36 | 5.36 |
| Bit length (um) | 0.67 | 0.67 |

Each Frame:

| | | |
|------------------|----|----|
| Subcode bytes | 1 | 1 |
| Data bytes | 24 | 24 |
| Q+P parity bytes | 8 | 8 |

| | | |
|-------------------|----|----|
| Total bytes/frame | 33 | 33 |
|-------------------|----|----|

Audio Data:

| | | |
|--------------------------|---------|---------|
| Audio sampling rate (Hz) | 44,100 | 44,100 |
| Samples per Hz (stereo) | 2 | 2 |
| Sample size (bytes) | 2 | 2 |
| Audio bytes per second | 176,400 | 176,400 |
| Sectors per second | 75 | 75 |

| | | |
|------------------------|-------|-------|
| Audio bytes per sector | 2,352 | 2,352 |
|------------------------|-------|-------|

Each Audio Sector (98 Frames):

| | | |
|------------------|-------|-------|
| Q+P parity bytes | 784 | 784 |
| Subcode bytes | 98 | 98 |
| Audio data bytes | 2,352 | 2,352 |

| | | |
|------------------------------|-------|-------|
| Bytes/sector RAW (unencoded) | 3,234 | 3,234 |
|------------------------------|-------|-------|

Hz = Hertz

mm = Millimeters

um = Micrometers

END OF APPENDIX A

APPENDIX B

Properties of Reed-Solomon Codes and Fatal Temporal Bit Drift

Reed Solomon codes are a subset of BCH codes and are linear burst (block) codes. A Reed-Solomon code is specified as RS (n,k) with s -bit symbols.

This means that the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol codeword. There are $n-k$ parity symbols of s bits each. A Reed-Solomon decoder can correct up to t symbols that contain errors in a codeword, where $2t = n-k$.

As described on Page 13 (repeated here for contiguity), the music CD Reed-Solomon code is RS $(255,223)$ with 8-bit symbols. Each codeword contains 255 code word bytes, of which 223 bytes are data and 32 bytes are parity. For this code:

$$n = 255, k = 223, s = 8$$

$$2t = 32, t = 16$$

The decoder can correct any 16 symbol errors in the code word such that errors in up to 16 bytes anywhere in the codeword can be automatically corrected.

Given a symbol size s , the maximum codeword length (n) for a Reed-Solomon code is $n = 2^s - 1$

The maximum length of a code with 8-bit symbols ($s=8$) is 255 bytes.

SUMMARY: This demonstrates how the code is created. It becomes obvious that the correction resolution limit is the size of the byte; or the “burst”. If a system requires deeper resolution, the code will require more space.

The CD and home computer evolved from an “8 bits to a byte” system, and so the resolution is based upon that size byte. Sub-byte resolution below that is not considered important; however, many double-blind, subjective tests indicate otherwise. See C.J. Huss, J. Gordon Holt, Larry Archibald et.al., in *Stereophile, The Highs and Lows of Double-Blind Testing*, May, 1985.

END OF APPENDIX B

APPENDIX C

Reed Solomon: Shortening of Code Blocks

Designers often opt to not use the actual sizes of Reed-Solomon code blocks. A technique known as “shortening” can produce a smaller code of any desired size from a larger code.

For example, the widely used (255,223) code can be converted to a (160,128) code by padding the unused portion of the block (usually the beginning) with 95 binary zeroes and not transmitting them. At the decoder, the same portion of the block is loaded locally with binary zeroes. More specifically, making a number of data symbols zero at the encoder, not transmitting them, and then re-inserting them at the decoder. The (255,223) code described above can be shortened to (200,168). The encoder takes a block of 168 data bytes, (conceptually) adds 55 zero bytes, creates a (255,223) codeword and transmits only the 168 data bytes and 32 parity bytes.

The compact disc is an example of an application of shortened Reed-Solomon codes. This shortcut is again predicated on the inaudibility of bit errors within a byte.

NOTE: Nova Physics is currently developing a series of recreations of nominal ‘Music CD’ codes (RS255/223) with and without shortening.

The idea is to examine if Bit Drift and Jitter changes take place when the 0’s are added subsequently, or not.

We hope to have results on this potentially VERY important data by June 30th, 2009, as it may be a critical area of degradation, which is often dismissed as numerically perfect.

END OF APPENDIX C

APPENDIX D

Reed Solomon: Finite Field Arithmetic

Reed-Solomon codes are based on a specialized area of mathematics known as Galois Fields or Finite Fields.

A finite field has the property that arithmetic operations (+,-,x,/ etc.) on field elements always have a result in the field.

Although it is a common misrepresentation in the HEA journals that Reed-Solomon is infinitely powerful and enjoys perfect correction, this is mathematics with nothing 'infinite' nor 'perfect'. Rather, we "halve the distance to the wall but never actually get to the wall itself", and therefore are subject to a percentage of efficacy. A Reed-Solomon encoder or decoder needs to carry out these arithmetic operations. These operations require special hardware or software functions (firmware usually) to implement.

Note: A Reed-Solomon decoder attempts to identify the position and magnitude of up to t errors (or $2t$ erasures) and to correct the errors or erasures.

END OF APPENDIX D

APPENDIX E

Reed Solomon: Imperfections in Perfection

Five Algorithms for Five Scenarios in Common Use: Syndrome Calculation, Symbol Error Locations, Error Locator Polynomial, Roots of the Polynomial, and Symbol Error Values

A Reed-Solomon codeword has $2t$ syndromes that depend only on errors (not on the transmitted code word). The syndromes can be calculated by substituting the $2t$ roots of the generator polynomial $g(x)$ into $r(x)$. This involves solving simultaneous equations with t unknowns.

Several fast algorithms are available to do this. These algorithms take advantage of the special matrix structure of Reed-Solomon codes, and greatly reduce the computational effort required. In general, two steps are involved:

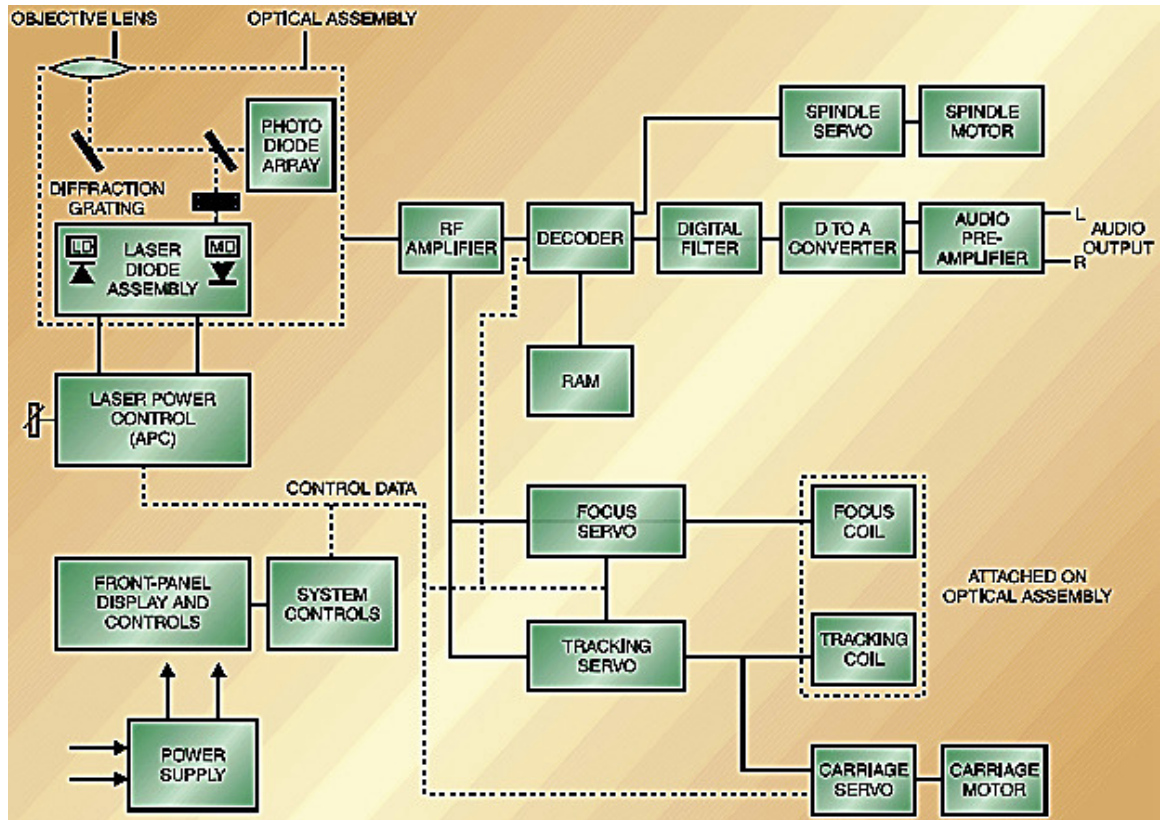
This can be done using the Berlekamp-Massey algorithm or Euclid's algorithm (yes Euclid, the more things change...). Euclid's algorithm tends to be more widely used in practice because it is easier to implement; however, the Berlekamp-Massey algorithm tends to lead to more efficient hardware and software implementation. This is done using the Chien search algorithm. Again, this involves solving simultaneous equations with unknowns. A widely-used fast algorithm is the Forney algorithm.

The Berlekamp-Massey algorithm is the best choice, generally. Euclid's is too generalized. Perhaps the 'best bang for the buck' is Symbol Parity Error Correctors, as only two bits are required to effect a 50% probability that an entire symbol will be perfected. While 50% is poor efficacy if nothing else were employed, it's still a very clever and highly efficacious outcome for the amount of space required. This is a similar calculation to parity calculation.

END OF APPENDIX E

APPENDIX F

Reading Pipeline & Areas of Vulnerability (Detailed Diagram)



END OF APPENDIX F