The SMPTE 27.97 fps drop-frame video time code system

Douglas A. Kerr

Issue 4
December 28, 2018

ABSTRACT

SMPTE time codes are used to identify specific "points" in a video recording or film to a precision of the time of one frame, the smallest unit by which film or video recordings can be edited, trimmed, and so forth. The time codes may be embedded in the video tape or film or in the digital representation of the "production" in an editor. Musical notation programs intended for use in scoring for video productions often use these time codes as a way of precisely coordinating the music with the video itself. The structure of the time codes there is dependent on the frame rate of the production.

A common frame rate for video recordings, following from the North American analog TV broadcast format in use since the advent of color television broadcasting, and still one of the options in the current digital television broadcast system, is nominally 29.97 fps (frames/second). With a non-integral number of frames per second, a straightforward time designation system (working in terms of integral hours, minutes, seconds, and frames) is not feasible. Rather, for this frame rate, a rather clever but tricky scheme is used. This article describes that scheme.

An appendix explains where this peculiar frame rate came from. A second appendix shows the details of the cyclic time discrepancy of the system. A third appendix describes an algorithm that can be used to convert time to hours:minutes:seconds;frames notation under this time code system. Other appendixes cover related matters, including the handling of time code matters in the musical notation program Overture 5.

The article only discusses this time code system from an abstract standpoint; there is no discussion of how the time code might be physically embedded in a film or video medium.

1 INTRODUCTION

Time codes are used in video production to precisely identify specific "points" in a video recording or film, generally to a precision of the time of one frame, the smallest unit by which video recordings can be edited, trimmed, and so forth. These codes are commonly used in the planning and execution of various video editing operations.

The time codes are typically presented for human interpretation in the format *hours*:*minutes*:*seconds*:*frames*, as for example 00:58:30:15.[1]

Music notation programs used in the composition and arranging of music for use in video or film productions often allow the video time code that would correspond to various spots in the score to be shown on the score and/or displayed in a clock-like window. This allows the proper coordination of the music with the video/film itself.

Various time codes of this general structure (accommodating different video/film contexts, notably different frame rates) have been standardized by the Society of Motion Picture and Television Engineers (SMPTE), and are often spoken of as "SMPTE time codes".

2 BEFORE VIDEO

Prior to the emergence of television broadcasting as a "parallel medium" to motion pictures, identification of points in a film production (during editing and such) was generally done on a *feet:frames* basis.[2] The emergence of television broadcasting, and especially the emergence of video recording, led to an interest in using a more time oriented basis, typically expressed in the form *hours:minutes:seconds:frames*. Then, of course, we need to become concerned not with the number of frames per foot of film but rather with the number of frames per second of wall time[3].

3 VIDEO FRAME RATES

3.1 U.S monochrome television transmission

The U.S. standard monochrome television system, standardized in 1941, utilized a nominal frame rate of 30 frames/second[4]. Some of the reasons for this are discussed in Appendix A.

Video time codes, as we know them today, didn't really exist in that era. There was no capability for recording video in the way we know of today, so there was really no need.

---

[1] We will see later that for the specific time code being discussed here, the preferred format would be 00:58:30;15.

[2] In 35 mm film production, for "counting" purposes it was assumed that 16 frames equaled one foot, even though in physical terms the actual relationship was a tiny bit different (the discrepancy being about 0.3%).

[3] My term for "real time"; the term of course comes from the notion of a wall clock.

[4] It is most common in industry practice to state this as "30 fps" (frames per second), but in this article I will follow accepted engineering and scientific practice and state the unit as "frames/second" or (abbreviated) "fr/s".

But if there had been time codes, the arrangements would have been very straightforward. If we think of a continuously-running "clock" carrying "time code time", the *frames* field would advance by one for every frame recorded. After the count got to 29, for the next frame the *frames* field would go to 00, and the *seconds* field would be incremented by one. And of course the *minutes* and then *hours* fields would advance just as we would expect.

3.2 U.S color television transmission

The U.S. color television system was standardized in 1953. For complicated reasons that are discussed in Appendix A, the frame rate was changed to (theoretically) 30•(1000/1001) frames per second (29.97002997). The time code system we will discuss here is predicated on a frame rate of 29.97 frames per second, which differs by only one part per million from that theoretical rate, and is convenient for the working of that time code system (as we will see shortly).

4 TIME CODE IMPLICATIONS AND THEIR SOLUTION

4.1 The complication

But now there is a complication in a time code system based on this new television frame rate.

We could of course use the same scheme as used with a 30 frames/sec frame rate–only the actual rate at which the *frames* field advances would be different (29.97 fr/s rather than 30 fr/s). But now, after a TV program lasting exactly 1 hour of "wall time", the time code "clock" would read 00:59:56:12, a discrepancy of 3.6 seconds. In production, this would be intolerable in terms of the coordination of program control and switching (not to mention in terms of revenue when advertising is selling for many thousand dollars per second).

Fancifully, this discrepancy could be avoided by having the *frames* field count up to only 29.97 frames before it cycled to zero and the *seconds* field was incremented. But of course the *frames* field must work in integer values (matching the advance of the actual frames of the video), so no such thing would be possible.

4.2 The solution

Instead, a clever scheme, somewhat reminiscent of how the leap year scheme of civil dates works, was devised. Here are its rules. Again, think in terms of a continuously-running "clock" carrying *time code time*.

- The *frames* field advances at the rate of 29.97 counts per second of "wall time" time (33.3666 ms per count).

- The range of the *frames* field is (normally) 00-29 frames. When the frames field reaches 29, at the next frame time the *frames* field goes to 00 and the *seconds* field is incremented by 1, except that:

- If the *seconds* field is now 00 (that is, we are now in the first second of a minute in **time code time**), the range of the *frames* field is 02-29 (that is, the frame numbers 00 and 01 are skipped at the beginning of that second), except that:

- If the *seconds* field is now 00, but the *minutes* field now is evenly divisible by 10 (that includes a value of 00), then the range of the *frames* field is the normal 00-29 (that is, no frame numbers are skipped at the beginning of that second).

The carries from the *seconds* field to the *minutes* field, and from the *minutes* field to the *hours* field, are conventional.

It is important to keep in mind that the seconds for which the first two frame values are skipped are the first seconds of minutes of **time code time** (not of **wall time**). This is an important distinction, since the two are slightly discrepant at this point.

The overall result of this pattern is that after 10 minutes of wall time the time code would be precisely 00:10:00:00–there is no discrepancy between the two at that point, nor at any other multiple of 10 minutes. And so after 60 minutes of wall time the time code would be precisely 01:00:00:00. [5]

But within any ten-minute period, there is a cyclic discrepancy between the time code and wall time in both directions. The greatest discrepancy in either direction is a little less than 2 frame times (about 0.07 sec).

This scheme is said to be a "drop-frame" time code scheme, an unfortunate term as it is of course not **frames** that are "dropped" but rather **frame numbers**.

5 Presentation

In a conventional time code scheme (not involving the "drop frame" complication), it is typical to present a time code value in this form:

00:58:30:15

[5] Note that if the actual frame rate were the specified nominal one (29.9700266), there would be a discrepancy at the end of 10 minutes of about 0.266 ms.

But for a drop-frame time code scheme (such as the one discussed here), it is the custom to use this format:

00:58:30;15

the semicolon before the *frames* field being a reminder that the *frames* field works in a "special" way.

Often in such contexts as the display of the time code in a digital panel, this convention is not observed, colons used for all separation of fields regardless of the time code system in use.

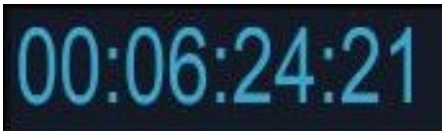
6 DESIGNATION

It is common to, as a shorthand, designate the time code system described here as "29.97 fps DF", the DF of course signifying "drop frame", a reminder of its special scheme of operation. For time code systems that do not use the drop-frame scheme, the suffix "ND" ("non drop [frame]") is often applied, especially when there is also a drop-frame form of the time code at the same frame rate so the two forms need to be unambiguously distinguished.

7 IN MUSIC NOTATION PROGRAMS

7.1 Functionality

In a music notation program being used to create or modify music intended for use in video productions, it is often possible to have the program place the time code values on the score itself, perhaps as of the beginning of every measure. We may also arrange for the time code time at any point in the score (perhaps at the nominal onset instant of a certain note we have selected) displayed on a visual "time code clock", as we see below with the notation program Overture 5 [6] .



Time code clock in Overture 5

If the score is not being "played", this is not happening in real time but in "conceptual time", reckoned for the score position of interest based on the tempo(s) prescribed for the music.

---

[6] Although I use Overture 5 for this example of a time code display, I must advise that in Overture 5 (as of version 5.5.1-6) there are numerous serious anomalies in its handling of time codes. Some of these are discussed in Appendix E.

7.2 A wrinkle in the calculation

In this situation there are of course no actual "frames" coming along. Rather, to show the time code applicable to any point in the score, we must first convert the musical position of that point to wall time. For example, if the meter of the score is 4/4 and the established tempo is a consistent 90 beats/minute, then the time of each measure would be 2.66̲ seconds.

Then, for the point of interest, we convert the corresponding wall time to measures, on the basis of 29.97 frames/second. The time code works in integral frames, so we must reduce this result to an integer. There are two ways to do this that would likely be considered:

- We truncate the result to an integer (that is, round the result **down** to an integer).
- We round the result to the **nearest** integer.

Clearly the choice is not of great import; the difference between the two approaches would be, in about half the cases, one frame, never greater. But let's still look at which would be more appropriate.

Conceptually, if we consider "playing" the film or video recording, one frame is in place for its allotted time and then the next frame is in place.[7]

Suppose we start "play" at the beginning of the film or video recording. We consider the wall time there to be "0", and we can consider the frame that is initially in place to be "frame 0", with time code 00:00:00;00. Only after one full frame time is the next frame ("frame 1") in place.

Thus it is appropriate for our reckoning of the time code to advance from 00:00:00;00 to 00:00:00;01 only after one full frame time. That is consistent with, when converting wall time to frames, **rounding down** the result of the calculation.

Thus, I endorse, while converting a position on a score in a music notation program into time code, when converting the reckoned wall time of the point of interest into frames, rounding down.

By the way, the use of rounding down has no significant effect on the range of time discrepancies encountered.

---

[7] In fact, with the "interlaced" display of video, that is not fully true, but I will still adopt that conceit for my presentation. It is essentially fully true for film viewing.

8 WHY NOT DROP ONE FRAME NUMBER EVERY HALF MINUTE?

In this system, the maximum discrepancy between the time code value and wall time is less than 2 frame times. Still, one might ask why we don't drop one frame number every half minute (instead of two every minute). This would cut the maximum discrepancy to about half of what it is now.

The reason is that in editing video, for a reason related to the modulation scheme of the NTSC color television system (see section A.2.5 in Appendix A), it is desirable in editing to, whenever possible, work with "clips" that contain an even number of frames, starting with an even-numbered frame.

Under a non drop-frame time code system, this is easily done by always working with clip boundaries that are between an odd and an even *frame number*. (The first frame of the whole thing is considered to be an "even" frame since it is "frame 0".)

Under the drop-frame system described here, this approach is still valid. Even and odd frames will still always have even and odd frame *frame numbers* in the time code, since when numbers are skipped it is always two at a time.

But that would no longer be true if the plan were to skip one frame number every half minute. So a complicated calculation would have to be made for each clip to determine where a desirable clip boundary was.

So the system doesn't do it that way.

9 THE APPENDIXES

Appendix A describes how the "peculiar" frame rate of U.S. color television transmission came about.

Appendix B illustrates shows, with detailed numerical information, the variations of the discrepancy between time code time and wall time.

Appendix C presents (in pseudocode) a routine for converting wall time into SMPTE 29.97 drop-frame time code values.

Appendix D discusses the 23.976 fps and 30 fps DF time code systems.

Appendix E Discusses the time code features of the notation program Overture (including some flaws in their implementation.

10 ACKNOWLEDGEMENT

Thank to Carla Kerr for her skilled proofreading of this tedious manuscript in an earlier version. Errors in this version are wholly my responsibility.

11 ISSUE RECORD

Issue 5 (this issue), December 28, 2018. Various editorial improvements. Corrections and extensive changes to Appendix B. New issue of algorithm in Appendix C.

Issue 4 (this issue), July 9, 2018. Correct discussion of rounding process. Correct minor typographical and editorial errors.

Issue 3, June 30, 2017. Make various corrections in Table 1 in Appendix B and in the discussion of it there. Add appendixes D and E. Correct minor typographical and editorial errors.

Issue 2, June 30, 2017. Correct minor typographical and editorial errors.

Issue 1, June 27, 2017. Initial issue.

-#-

Appendix A
Origin of the "29.97" frames/second rate

A.1 U.S. monochrome television transmission format

As the prospect of television broadcasting (monochrome at this point in time) emerged in the late 1930s, various schemes of modulation and image formatting were used on an experimental basis. Ultimately, one scheme was adopted as the premise for U.S. television broadcasting, and its particulars were refined and standardized by the National Television System Committee (NTSC), a standards body established by the Federal Communications Commission for television broadcasting in the U.S. The resulting standard was adopted by the FCC in early 1941 as the norm for all U.S. television broadcasting.[8]

In this scheme, the image was turned into what would later come to be described as a "video signal" by scanning the image in a raster pattern. Successive instances of the entire image were scanned 30 times per second, said to be the *frame rate*. The frame consisted of 525 horizontal *scan lines*.

Actually, each frame was conveyed by two successive fields, each of which carried either the odd-numbered or even-numbered lines of the frame raster; thus the *field rate* was 60 fields per second. This arrangement was in the interest of minimizing the visual impression of flicker in the displayed image.

Transmission was over a radio-frequency carrier in the VHF band. The video signal, representing the a non-linear transform of the luminance of the point in the image[9], was carried by vestigial sideband amplitude modulation of this carrier. For each channel, the main carrier was accompanied by an audio carrier 4.5 MHz higher in frequency. The audio aspect of the program was carried by frequency modulation of that second carrier.

The choice of 60 fr/s was made to mitigate the impact of a common source of interference in the received signal. Nonlinearities in electrical appliances or lighting fixtures (especially the then-emerging fluorescent fixtures), or in defective joints in power transmission line (or even household wiring) conductors, would generate harmonics of the power line frequency potentially extending into the radio-frequency band used for television transmission. These harmonics appeared to be amplitude modulated at a frequency of nominally 120 Hz (twice the power line frequency–the

---

[8] Although this format was standardized by the NTSC (in its first incarnation), it is never spoken of as the "NTSC television format standard", that term by custom being reserved for the color television format standardized by a reincarnation of that same body a number of years later.

[9] But still often called *luminance*.

emissions would normally be at a maximum for both positive and negative excursions of the instantaneous voltage).

Thus there would be in the demodulated video signal a spurious component recurring at a rate of nominally 120 Hz. As this rate corresponds nominally to twice the field rate of the television signal (which itself is two times the frame rate), this would typically produce two shaded bars across the image in the television receiver, nearly stationary vertically.

If the frame rate of the TV signal was not nominally equal to the power line frequency, these bars would move vertically across the picture at a substantial rate, and would thus be much more annoying visually than the nearly-stationary bars that resulted from the use of the 60 fr/s frame rate.

A.1.1 In Europe

In Europe, where the most common power-line frequency was 50 Hz, the (monochrome) television system that was most widely adopted used a nominal frame rate of 50 frames/second, based on the very same reasoning.

A.2 U.S. color television transmission format

A.2.1 Introduction

During the late 1940's there was earnest interest in the prospect of "color" television broadcasting. As with television broadcasting itself, various systems were devised, tested, and promoted by various manufacturers. Eventually a system developed by RCA was adopted as the premise for the introduction of color television broadcasting in the U.S.

The FCC reincarnated the NTSC, under whose auspices the particulars of the color TV broadcast system were refined and standardized. The resulting system was adopted by the FCC in 1953. This system came to be known as the "NTSC" system (notwithstanding the fact that the earlier standard monochrome system was also perfected under the auspices of the NTSC in an earlier incarnation).

A.2.2 Forward- and backward compatibility

An important property of the NTSC system was forward- and backward-compatibility between the monochrome and color systems, meaning:

- Existing monochrome receivers tuned to a color broadcast would properly display the picture (on a monochrome basis, of course).

- Color receivers tuned to a monochrome broadcast would properly display the picture (on a monochrome basis, of course), without needing to make any significant mode change.

A.2.3 The modulation scheme

In the NTSC system, the payload of the vestigial-sideband amplitude modulation of the main carrier is the *luma* signal[10], plus a subcarrier (at a frequency of about 3.58 Mhz) which, by suppressed-carrier phase-amplitude modulation,[11] conveys the quasi-chrominance[12,13] of the points in the image. This new component came to be called the *chroma* signal (in part to recognize that its payload is not actually chrominance).

A.2.4 Minimization of undesirable artifacts

The chrominance subcarrier (yes, it was most often called that even though its payload is not truly chrominance[14]) and its sidebands share part of the same frequency region occupied by the luma signal itself. Thus, the luma and chroma signals are imperfectly separated. The result is that there are visual artifacts from the chrominance subcarrier (especially in the case of a monochrome receiver receiving a color transmission). In effect, the receiver misinterprets the chroma signal as part of the luma signal.

It was recognized that the visual impact of these artifacts could be minimized if certain relationships obtained between the horizontal scan rate, the chrominance subcarrier frequency, and the separation between the main carrier and the audio carrier. For various reasons, the precise intercarrier separation was considered sacrosanct.

That being the case, attaining the desirable relationships required a slight change in the horizontal scan rate and thus (since the number of scan lines per frame was also considered sacrosanct) a slight change in the frame rate.

The final design has a frame rate that theoretically was 1000/1001 times the original frame rate of 30 fr/s. That theoretical value is 29.97002997 fr/s. For various practical reasons, the actual specified frame rate was made 29.97002667 fr/s, with a tolerance of ±0.000088 fr/s. (The theoretical value is well within that tolerance.)

---

[10] Something like the non-linear transform of luminance used in the monochrome system, but not exactly, and here called "luma" to make clear that it was not luminance.

[11] This is also commonly described as *quadrature amplitude modulation*; the two terms describe the same situation from two different perspectives.

[12] Chrominance refers to the "component" of a color (in the formal sense of that term) that is responsible for its being "colored" (in the popular meaning of the term).

[13] This is not true chrominance but is a non-linear property related to chrominance in a complicated way.

[14] In later years it came to be called the "chroma signal" to make clear that it did not convey chrominance.

In order to play the “drop-frame” time code game and have wall time and time code time come exactly together over a cycle of reasonable length, the time code scheme was predicated on a frame rate of exactly 29.97 fr/s. The disparity between this and the theoretical or specified rates is less than one part per million.

A.2.5 Editing considerations

One important feature of the plan regarding the chrominance subcarrier frequency and the frame rate is that the base phase of the subcarrier is opposite in successive frames. This minimized the visual impact of artifacts resulting from the fact that the receiver misinterprets the chroma signal as part of the luma signal.

Because of this situation, when editing NTSC video recordings, it is desirable to always make “cuts” of sets of frames that start with an “even” frame and contain an even number of frames. This maintains continuity of the chrominance subcarrier base phase across the “cut”. This had an influence on the design of the SMPTE 29.97 fr/s drop-frame time code system, as discussed in section 8 of the body of this article.

A.2.6 In Europe

As a standard for color television transmission emerged in Europe, one camp supported a concept quite like the NTSC system used in the U.S., but were aware of performance limitations of that system. (Workers in the field used to jokingly say that NTSC stood for “Never Twice the Same Color.) Thus the system that eventually became the most widely used in Europe, PAL (“Phase Alternation Line”), included some important design differences (“improvements”) from the NTSC system. The final scheme did not demand any tampering with the frame rate, which remained at (nominally) 50 fr/s.

Thus the time code for use with PAL-oriented video is straightforward, no “drop-frame” scheme being needed.

-#-

Appendix B
Time discrepancy in the 29.97 fr/s drop-frame time code system

Under the 29.97 fr/s drop-frame time code system, the frames come along at a rate of 29.97 per second, but the time code clock only scores a new second after 30 frames have passed. Thus the time code drops behind "wall time" by 0.03 frame time each second. But we give the time code a two-frame "bump" at nine out of the ten "new minute" instants in each ten-minute period, which exactly overcomes this discrepancy.

If we want to be more specific, and ask, for example, "how much is the greatest discrepancy", we must first address several issues, including:

- Will we look into the discrepancy at each integer frame instant, or at each integral second of wall time, or what?
- Just exactly what two things will we compare to score the "discrepancy"?
- Will we reckon the discrepancy in seconds or frames?

In this presentation I chose as follows, based on the "frame-oriented" nature of this whole matter:

- I will reckon the discrepancy at certain selected integral frame counts.
- I will reckon the discrepancy in frames (or frame times, if you will).

As to the discrepancy, I will compare the following:

- The total frames implied by "time code" that would be shown at that instant.
- The wall time at that point (on a frame basis).

And thus the discrepancy will be denominated in frames.

Table 1 shows the pattern of time discrepancy under this rubric. The entries are all instants at "interesting" integral numbers of frames from the starting instant. For each we see the wall time and the corresponding time code.

Instant (frames)	Wall time[1] (hh:mm:ss:ff.ff)	Time code (hh:mm:ss;ff)	Discrep-ancy (frames)	Note
0	00:00:00:00.00	00:00:00;00	0.00	Starting instant
1	00:00:00:01.00	00:00:00;01	0.00	Time code and wall time are fully consistent
2	00:00:00:02.00	00:00:00;02	0.00	
3	00:00:00:03.00	00:00:00;03	0.00	
27	00:00:00:27.00	00:00:00;27	0.00	Still consistent
28	00:00:00:28.00	00:00:00;28	0.00	Still consistent
29	00:00:00:29.00	00:00:00;29	0.00	Still consistent
30	00:00:01:00.03	00:00:01;00	−0.03	Time code falls behind
59	00:00:01:29.03	00:00:01;29	−0.03	
60	00:00:02:00.06	00:00:02;00	−0.06	Time code falls behind
899	00:00:29:29.87	00:00:29;29	−0.87	
900	00:00:30:00.90	00:00:30;00	−0.90	Time code falls behind
1799	00:01:00:00.80	00:00:59;29	−1.77	Time code fell behind in 1st minute
1800	00:01:00:01.80	00:01:00;02	+0.20	Frame numbers 00, 01 skipped[2]
3597	00:02:00:00.60	00:01:59;29	−1.57	Time code fell behind in 2nd minute
3598	00:02:00:01.60	00:02:00;02	+0.40	Frame numbers 00, 01 skipped
5395	00:03:00:00.40	00:02:59;29	−1.37	Time code fell behind in 3rd minute
5396	00:03:00:01.40	00:03:00;02	+0.60	Frame numbers 00, 01 skipped

Start of minutes 4-8 not shown				Frame numbers 00, 01 skipped five times at minutes 4-8
16183	00:08:59:29.17	00:08:59;29	−0.17	Time code close here
16184	00:09:00:00.20	00:09:00;02	+1.80	Frame numbers 00, 01 skipped
17981	00:09:59:28.97	00:09:59;29	+0.03	Time code almost exact here
17982	00:10:00:00.00	00:10:00;00	0.00	No frame numbers skipped

Notes

1. In which the seconds counter advances by one every 29.97 frames counter and the frames field reads to 0.01 frame.

2. This is triggered by the actual, not theoretical, time code reaching what would be 00:01:00;00 before the "skipping".

Table 1. Time discrepancy in the 29.97 fr/s drop-frame time code system

The column headed "Wall time" shows the time at the instant of interest in terms of hours, minutes, seconds, frames, and hundredths of a frame. Note that in this mixed-base form, with the number of frames per second being non-integral, as time progresses, just when the frames component would reach 29.97, it instead goes to 00.00 and the seconds field is incremented.

We start by looking at the operation at early frame counts (within the first second). We note that for each additional frame that passes, the time code advances by one frame (as does the wall time). Thus there is no discrepancy between the two times in this season.

Next, we go to the end of the first second. As the frame count reaches 27, 28, and 29 the time code and wall time are still together. At a frame count of 30, the time code advances its seconds field, now showing 00:00:01;00. But at that instant, the wall time is 00:00:01:00.03. Thus the time code is 0.03 frames behind wall time.

We next move forward by about one minute. The frame count when the time code "would show" exactly 1 minute (except for the skipping of frame counts that will occur then) is 1800 frames. We will first look at things one frame earlier, at a frame count of 1799.

We see that there the time code has fallen behind the wall time by 1.77 frames.[15]

> It's a little tricky to confirm "at sight" the discrepancy by subtracting the time code from the wall time. In this case, to subtract the "29" value of the time code frames field from the "00.80" value of the frames field of the wall time, we have to borrow from the seconds field of the wall time, and one second is worth 29.97 frames. Thus the wall time, after "borrowing" one second, is 00:00:59:30.77. We subtract the time code, 00:01:59:29, from that, and get 1.77 frames (but since the time code is the smaller, that is actually a discrepancy of -1.77 frames). This same maneuver will be needed at some other places.

At 1800 frames, when the time code "turns over" to the next second (and would, except for frame number skipping, then show exactly one minute), the time code jumps ahead by 2 frames (since frame numbers 00 and 01 are skipped), and thus the wall time code is now 0.20 frames ahead of the theoretical time code. (There is no shift of the discrepancy by 0.03 frames at this time—that is only seen at a new frame time when the seconds field of the wall time advances, which it does not here.)

The frame count when the time code clock "would show" exactly 2 minutes (except for the skipping of frame counts that will occur there) is 3598 frames [16]. We will first look at things one frame earlier, at a frame count of 3597 frames. We note that the discrepancy here is only –1.60 frames, 0.2 frames less (in magnitude) than we had approaching the "1-minute mark". That is because at the 1-minute mark, the two frame number skip was a little more than needed to overcome the amount the time code had fallen behind during the first minute.

In any case, we then look at the situation at 3598 frames. The action here is parallel to what we saw at the "1 minute" point. (Again there is no shift of the discrepancy by 0.03 frames here since the seconds field of the wall time has not advanced.)

We see the further progression of this pattern at the "3-minute" point (5396 frames). (Again there is no shift of the discrepancy by 0.03 frames here since the seconds field of the wall time has not advanced.)

This process proceeds this way over the successive one-minute intervals. At the end of each, the amount the time code is behind is less and less, because at the start of each new time code minute the time code advance

---

[15] We will see that this is the greatest negative discrepancy (that is, when the time code is farthest behind the wall time for an integral frame time).

[16] The "should be 2 minutes" time code point does not come at 3600 frames but 2 frames earlier since two frame counts were "jumped" at the "1 minute" point.

due to skipping the two frame numbers is greater than the amount the time code has inherently fallen behind during the preceding minute.

Now that we have seen the way the situation progresses, we will jump ahead to the "9-minute" point, which occurs at frame 16184. One frame before that (frame 16183) we see that the discrepancy has become only -0.17 frame times, and one frame later, the 0.03 frame progression in discrepancy (which occurs whenever the wall time seconds field advances upon a new frame) plus the two-frame skip at frame 16184 puts the discrepancy to +1.80 frame times.[17]

At the 17981 frame point (one frame before when the time code clock would show exactly 10 minutes, and it in fact will), the time code has drifted back so it is 0.03 frames ahead of the wall time. At 17982 frames (the "10-minute point"), the time code has encountered its usual 0.03 frame loss because the wall time seconds count advances here; no frame counts are skipped (as this is the beginning of a minute evenly divisible by 10); and the two time scales come into exact conformity.

This whole scenario plays out over every ten-minute cycle (that is of course both exactly 10 minutes of wall time and exactly 10 minutes of time code time).

For integer frame counts, the greatest negative discrepancy between the time code and the wall time is −1.77 frame times, which occurs at the end of the first minute of each 10-minute cycle. The greatest positive discrepancy is +1.80 frame times, which occurs at the "9-minute mark" in each 10-minute cycle.

If we examine the time discrepancy at other times, including various integer seconds of wall time (I will spare the reader that presentation), we find that the magnitude of the maximum discrepancy can be slightly larger than 1.80 frame times but still not ever over 2 frame times.

-#-

[17] This is the greatest positive discrepancy (that is, when the time code is ahead of the wall time for an integral frame time).

Appendix C
Algorithm for converting "wall time" to 29.97 fps drop frame time code

C.1 Introduction

This appendix presents, in the author's form of pseudocode, an algorithm for converting an instant described in terms of "wall time" (time in seconds from some reference point)) into the corresponding time code under the SMPTE 29.97 fr/s drop-frame time code scheme.

C.2 Caveat (in the expected fine print, of course)

Although the author has used his best skill to formulate this algorithm, he does not guarantee its accuracy nor suitability for any purpose. Readers who may wish to use this algorithm as, for example, the basis for program code in an application should first confirm that it is suitable. Any adoption of this algorithm is done at the person's sole discretion and risk, and the author cannot be responsible for any results deemed unsatisfactory.

C.3 The algorithm

Routine for converting a "wall time" (in seconds) into SMPTE time code form under the 29.97 fps drop frame system

Issue 05, 2018.12.27

Author: Douglas A. Kerr

Language: DAK pseudocode

Issue record

Issue 05 (this issue), 2018.12.27. Editorial changes, change and correction in variable name, no actual code changes.

****

// Define constants

frameRate = 29.97

sizeBigCycle = 17982 // Size in frames of the "ten minute" cycle of life of the time code system

// This cycle corresponds precisely to a wall time of 00:10:00.00 and a time code of 00:10:00;00

// No frame counts are skipped at the beginning of a big cycle.

sizeWeeCycle = 1798 // Number of frames between points where frame counts might be skipped. The skip occurs at the beginning of the wee cycle.

// This interval is approximately one minute of wall time and there are ten of them in a big cycle

// However, the first wee cycle in a big cycle actually has a duration of sizeWeeCycle +2 since no frame numbers are skipped at the beginning of minute 0 (the start of timekeeping) because the minute number there is evenly divisible by 10.

```
// Declare variables

wallTime  //Input: wall time in seconds at the instant of interest

timeCodeHMSF  //Output: time code in H:M:S;F form at the instant of interest

frames1     // Number of frame times to the instant of interest

numBigCycles            //Number of full big cycles

tailFrames      // Number of frame times in the "tail" (that is, after all full big cycles have
      been removed)

numWeeCycles     // Number of partial or full wee cycles in tail

numSkips1     // Number of times in the tail that frame counts are skipped

numSkips2     // Number of times in all the full cycles that frame counts are skipped

numSkips3     // Total number of times that frame counts are skipped

framesSkipped     // Total number of frame counts skipped

adjustedFrames          // Total number of frames to the instant of interest as will be
      reported in the time code


// Declare function

HMSF(numFrames)  // Converts argument numFrames (in frames) to H:M:S;F form (30 frame
      counts = 1 seconds count).

      // This function not defined here. Algorithms for it are well known.


// START

frames1 = integer(wallTime * frameRate)     // Convert wall time to frames, truncate result
      to integer. (See section 7.2in the body of the article.)

numBigCycles = integer(frames/sizeBigCycle)      // Determine number of full big cycles in
      frame count.

tailFrames = frames - (numBigCycles * sizeBigCycle)     // Number of frames in tail (beyond
      all full big cycles); might be zero.

if (tailFrames < (sizeWeeCycle + 2))      // First wee cycle is 2 frames larger than all later
      ones

      numWeeCycles = 1     // If tailFrames = 0 this will work out properly

else

      numWeeCycles = integer ((tailFrames - 2) / sizeWeeCycle)  +1     // Number of wee
      cycles (full or partial) if over 1.

      // Note that when a full wee cycle has just been counted as completed, we are at the
      first frame position of the next wee cycle (after any skipping that occurs at the
      beginning of that wee cycle)..

        // The -2 accounts for the fact that the first wee cycle has length sizeWeeCycle +2.

numSkips1 = numWeeCycles -1     // Number of times in tail that frame counts are skipped
```

// The -1 takes into account that no frame counts are skipped at the beginning of the first wee cycle, which falls at a time code minute whose number is an integral multiple of ten. Also, if tailFrames = 0, numWeeClycles will be calculated as 1, where as actually there are no wee cycles in the tail. And numSkips1 will be properly calculated as 0.

numSkips2 = numBigCycles * 9 // Number of times in all the full big blocks where frame counts are skipped

numShips3 = numSkips1 + numSkips2 // Total number of times where frame counts are skipped.

framesSkipped = numSkips3 * 2 // Total number of frames skipped

adjustedFrames = frames1 + framesSkipped // Calculate "adjusted frames" by applying all frame skips; a frame skip is a “jumping ahead” of the time code time.

timeCodeHMSF = HMSF(adjustedFrames) // Convert adjusted frames to H:M:S;F form.

// This is the result.

// END

-#-

Appendix D
The 23.976 fr/s and 30 fr/s DF time codes

D.1 Introduction

Two other "odd" time code systems exist as creatures of the same context that spawned the 29.97 fr/s DF time code. I will discuss them in this appendix.

D.2 Caveat

I do not work in the video or film industries, and information on the time codes discussed in this appendix is hard to come by. What I describe here has been extracted by much reading between the lines. Readers who know better are encouraged to being their knowledge to my attention.

D.3 The 23.976 fr/s time code system

D.3.1 Introduction

During the era of the "monochrome" TV broadcast standard in North America, existing motion picture were a popular source of television programming. But a clever scheme had to be employed to harmonize the 24 fr/s frame rate of the motion pictures with the (then) 30 fr/s frame rate of television broadcasting.

D.3.2 The field stricture of the television signal

Before we look into the actual working of the 23.976 fr/s time code system, we must review a previously-unmentioned detail of the NTSC color television broadcast system.

We have spoken of the transmission of 30 frames/second, but the actual process is a little more complicated. The actual system transmitted 60 images, called *fields*, per second. First, a field would be transmitted that carried every other line of the scanned whole image (say, the "off-numbered" lines. Then a field would be transmitted that carried the "even-numbered" lines of the scanned whole image. The object was to minimize the visual sensation of flicker by doubling the rate at which new "images" were presented on the screen. Because the two sets of lines were "interlaced" on the screen, this scheme became known as "interlaced transmission".

D.3.3 Reconciliation of the frame rates

Now to the actual matter of delivering a film with a frame rate of 24 fr/s over the television system. Conceptually, one frame of the film is "scanned into" 2 consecutive fields. The next frame of the film is scanned into 3 consecutive fields. In the "2-field" case, these might or might not be the field of a single television frame. Half the time they would be, and half the time not. In any case, on the average, each film frame is transmitted in the time of 2.5 television fields, or 1.25 television frames, perfectly reconciling the

24 fr/s rate of the film to the 30 fr/s television rate. Looking at it another way, each four frames of the film are presented by five television frames. But in the "classical" way of doing it (as described above), two of those five video frames contain scan lines from two different film frames (in their two fields). But the whole thing is very satisfying to the human eye.

This scheme is often called the "3:2 pulldown" system,[18] the term *pulldown* being borrowed from film technique in which, in a camera or projector, film was periodically "pulled down" from frame to frame (at the film frame rate).[19]

D.3.4 The impact of the new television frame rate

A fly came into this ointment with the development of the NTSC color television system, with its frame rate of (nominally) 29.97 fr/sec. Existing films were transmitted with the same scheme described just above, but were run at an average frame rate of 23.976 fr/s. This of course meant that the action was 0.1% slow (hardly noticeable), and the sound ended up with all its components 0.1% lower in frequency than originally intended (again hardly noticeable[20]).

Another small housekeeping problem is that a film with a run time of exactly 90 minutes would take 90 minutes plus 0.54 seconds to deliver in the television signal.

But, in order to avert this small discrepancy in films newly shot with the intent that they be presented on television, those films were sometimes shot at a frame rate of 29.976 fr/s. When transmitted on a "3:2 pulldown" basis on a television system with a frame rate of 29.97 fr/s, the average advance of the film was 19.976 fr/s, the exact rate at which it was shot. Thus there would be no slowing down of the action, and no pitch shift to the sound.

D.3.5 A time code for this

When such a film is being edited, the time code used is called the 23.976 fr/s time code. It works on the basis of 24 frame counts per "seconds count", although of course each "seconds count" is worth 1.001 seconds of real time.

---

[18] Given that it is customary to speak of the first frame being scanned into 2 fields and the second frame into 3 fields, we might think that the notation would be "2:3 pulldown".

[19] But the term "pulldown" is also used to refer to playing a video recording (or film) at a "slightly" slower speed than that at which it was recorded, in order to accommodate various forms of transfer; the frame rate is "pulled down"!

[20] This is a pitch shift of about 1.7 cents; the *cent* is a unit of musical pitch difference such that a pitch difference of 100 cents corresponds to a pitch difference of one semitone.

The result is that the time code time runs slower than "real time" by 0.1%.

Now, couldn't that discrepancy have been averted by the adoption of a "drop frame" scheme at this frame rate? Sure. One that has been used (but was never "standardized") is actually predicated on a frame rate of exactly 23.98 fps rather than 23.976 fps[21]. It is sometimes called "23.98 fps True Time".[22]

The frames count is 24 per second (the frames element running from 0-23), except that as we approach each 50 second multiple of time code "time" (*e.g.*, 00:00:50:00, 00:01:40:00, 00:02:30:00, etc.), after the frames count reaches 22 (*e.g.*, 00:00:49:22), frame count 23 is skipped, and at the next frame the time code is the exact minute (00:00:50:00).[23]

Thus, overall, the time code time remains consistent with "real time", the two coming precisely together every 50 seconds. The maximum discrepancy is 1 frame (the time code being behind "real time").

A very similar scheme could be used with an actual frame rate of 23.976 fps (but has never been standardized). It works exactly as described just above, except that in addition to the rules as stated there, as we approach each 250 second multiple of time code "time", after the frame count reaches 21, frame counts 22 and 23 are skipped, and at the next frame the time code is the even second.

Again, ongoing, the time code time remains consistent with "real time", the two coming precisely together every 250 seconds. The maximum discrepancy is 1 frame (the time code being behind "real time").

Still, in most cases, the straightforward 23.976 fps time code system. "running slow by 0.1%", is used.

D.3.6 Digital television

Of course, television broadcast under the NTSC system itself is all but obsolete in North America, being superseded by the digital television broadcast system. There is a complicated set of standards embracing a variety of formats, administered by the Advanced Television Systems Committee (ATSC). Many of these formats provide for imaging at a higher definition than that of the NTSC system (often referred to as "HDTV"—high

---

[21] In fact, often the 23.976 fps frame rate is spoken of as "23.98 fps", and in many cases the nominal operating frame rate is actually 23.98 fps.

[22] My information on this has been deduced in a very indirect way, and cannot be considered authoritative.

[23] This happens every 1199 frames.

definition television). These formats include a range of different frame rates and frame structures. One frame rate that is included (for reasons of continuity) is 29.97 fr/sec. This is like life imitating art.

So we still have need for the 29.97 fr/s drop-frame time code system, now usually for video recordings rather than actual film.

D.4 The 30 fr/s drop frame time code system

In some cases, a video production intended for television broadcast will be created at a frame rate of 30 fr/s. If it is broadcast in a 29.97 fr/s context, it will have to be "run slow" by 0.1%.[24]

In the early stages in its life, it will be edited on a 30 fr/s basis. But later in its life, it may be edited on a 29.97 fr/s basis. And in the latter phase, the 29.97 fr/s drop frame time code system will probably be used.

So, in some situations, so that there would be a frame-by-frame correspondence in time codes between the two editing contexts, in the 30 fr/s context, the same drop-frame scheme is used as is used in the 29.97 fr/s drop frame time code system. That is, every time the time code enters a new minute (except for minutes whose number is a multiple of 10), frame counts 00 and 01 are skipped). This makes the time code clock, overall, run "0.1% fast" compared to real time.

It is the existence of this odd creature that is responsible for the ordinary 30 fps time code system sometimes being spoken of as "30 fps NDF ("non drop-frame").

-#-

[24] And today there are sophisticated sound processing systems that can make a "pitch shift" to overcome the pitch shift that would otherwise result from this situation.

Appendix E
Time code features of the notation program Overture

E.1 Introduction

Overture is a very sophisticated full-featured music notation program, published by Sonic Scores. Inc., and developed on an ongoing basis by Don Williams of Sonic Scores.

E.2 Features for film and television music scoring

The program offers several features of special interest to those writing or arranging music intended to be used in film or video productions. Among these are features relating to SMPTE time codes. Those features include:

- Showing where the cursor is in the score on a "little clock" in time code form (reckoning this is of course predicated on the tempo established for the score).

- Indicating on the score, perhaps at the beginning of every system, perhaps at the beginning of every measure, the time code there (again predicated on the tempo established for the score).

These features can operate with the user's choice of one of several time code systems:

- 23.976 fr/s

- 24 fr/s [For work with film or film-compatible video in a non-television context]

- 25 fr/s [For work in the context of the European television system]

- 29.97 fr/s drop frame [For work in the context of the North American broadcast television system].

- 30 fr/s non-drop frame (Common for video in a non-broadcast-television context_.

E.3 Errors in execution

But, according to tests and analyses made here[25], there have been a number of serious errors in Overture's execution of the features. Among those are:

- For the non-integer frame rate systems (the 23.976 fr/s and 29.97 fr/s DF systems), the rate of advance of the frames "counter" is not that of the time code system, but rather is 24 fr/s and 30 fr/s, respectively.

---

[25] For example, on Overture test version 5.5.1-6.

- For the 29.97 fr/s drop frame system, the frame number dropping is not done correctly. Two frame numbers are skipped at the one-minute point, but none thereafter.

There were other, more subtle anomalies, which I will not discuss here.

E.4 Improvements

As of this writing (in Overture 5.5.4-3), two of these major anomalies have been corrected. The frame advance rate for the 29.97 fr/s drop-frame system is now correct, and the drop-frame scheme operates correctly. But other related anomalies have now appeared.

E.5 Interaction with the developer

This office has repeatedly called these apparent errors to Don Williams's attention (in considerable detail, with illustrative examples). The only acknowledgement received was the revocation of the author's membership in the forum where problems with Overture 5 are discussed..

-#-