The SMPTE 27.97 fps drop-frame video time code system

Douglas A. Kerr

Issue 3
June 30, 2017

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 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 coordinating the music with the video itself. The structure of the time codes 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, is nominally 29.97 fps (frames/second). We can readily see that 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 tricky scheme is used. That scheme is described in this article.

An appendix explains where this peculiar frame rate came from. A second appendix shows the details of the time discrepancy of the system. A third appendix describes an algorithm that can be used to convert actual time to hours:minutes:seconds;frames notation under this special time code system.

The article only discusses this time code system from an abstract standpoint; there is no discussion of how the time code is 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 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 clock-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 usual format would be 00:58:30;15.

[2] In 35 mm film production, for "counting" purposes it was assumed that there were 16 frames to the foot, even though in physical terms the relationship was a tiny bit different.

[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 "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 work 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 advantageous to 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.

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 we need the *frames* field to 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 (basically) 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 00 (that is, we are 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 00, and the *minutes* field is evenly divisible by 10 (that includes a value of 00), then the range of the *frames* field is 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).

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 at that point, not 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. And the greatest discrepancy within that period is about 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

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.

6. DESIGNATION

It is common to, as a shorthand, designate the time code system described here as "29.97 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).

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.

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 2.66$\underline{6}$ seconds.

Then, for the point of interest, we convert the corresponding wall time to measures, on the basis of 29.97 measures/second. The time code works in integral measures, 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.[5]

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.

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.)

---

[5] In fact, with the traditional "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.

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.

Error! Reference source not found. 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.

10. ACKNOWLEDGEMENT

Thank to Carla Kerr for her skilled proofreading of this tedious manuscript.

11. RELEASE NOTICE

This issue is released to correct several minor typographical errors (my proofreader was overloaded at the time of the previous release so it went up without her attention).

#

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.[6]

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 60 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 120 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 quasi-luminance[7] of the point in the image, was carried by vestigial sideband amplitude modulation of this carrier. For each channel, the main carrier was accompanied by an audio intercarrier 4.5 MHz higher in frequency. The audio aspect of the program was carried by frequency modulation of that intercarrier.

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

---

[6] 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.

[7] I call it that because it does not actually carry the luminance of the image points but rather, for reasons that are beyond the scope of this article, a non-linear transform of that property.

for television transmission. These harmonics appeared to be amplitude modulated at a frequency of nominally 120 Hz (twice the power line frequency–the 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 anointed 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 standard monochrome system was also perfected under the auspices of the NTSC).

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 a quasi-luminance signal [8], plus a subcarrier (at a frequency of about 3.58 Mhz) which, by suppressed-carrier phase-amplitude modulation,[9] conveys the quasi-chrominance [10,11] 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 is most often called that even though its payload is not truly chrominance) 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 between the horizontal scan rate, the chrominance subcarrier frequency, and the separation between the main carrier and the audio intercarrier obtained. For various reasons, the audio 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

---

[8] Different from the quasi-luminance signal used in the monochrome system, and often called “luma”, in part to recognize that its payload was not actually luminance.

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

[10] 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).

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

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.)

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 result 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. That was in fact one of the objectives of the plan. 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 has an influence on the design of the SMPTE 29.92 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 frames/second.

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

#

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 0.030 frame times at the end of each second. But we give the time code a two-frame "bump" each minute for nine out of the ten minutes boundaries 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?
- Will we reckon the discrepancy in seconds or frames?
- Just exactly what two things will we compare to score the "discrepancy"?

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 exact frame counts.
- I will reckon the discrepancy in frames (or frame times, if you will).

That leaves the question as to just what things will we compare to determine the exact discrepancy at that frame count. Here, I will compare the following:

- The actual time code that would be assigned to the frame count of interest.
- The "time code" shown by a fanciful time code clock that indeed scores a new second after the passage of 29.97 frames in the current "second", and which further shows us the *frames* value to a precision of 0.01 frame. I will call this time representation the "theoretical time code".

Of course if we actually made such a thing it would not be at all handy, since (after the first second) the frames field would not advance in synchrony with the arrival of the frames themselves. But it is a useful conceit for this presentation.

We can, by the way, think of the "theoretical time code" as being a representation of the wall time in an odd mixed-non-integral-base form.

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 corresponding theoretical time code to the nearest 0.01 sec. and the actual time code value.

Instant (frames)	Theoretical time code (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	0.000	Starting instant
1	00:00:00:01.00	00:00:00;01	0.000	Time code and theoretical time are fully consistent
2	00:00:00:02.00	00:00:00;02	0.000	
3	00:00:00:03.00	00:00:00;03	0.000	
29	00:00:00:29.00	00:00:29;00	0.000	
30	00:00:01:00.03	00:00:01;00	−0.030	Time code falls behind
1799	00:00:59:29.77	00:00:59;29	−1.770	Time code fell behind in 1st minute
1800	00:01:00:01.80	00:01:00;02	+0.200	Frame numbers 00, 01 skipped
3597	00:02:00:00.60	00:01:59;29	−1.570	Time code fell back in 2nd minute
3598	00:02:01:01.60	00:02:00;02	+0.400	Frame numbers 00, 01 skipped
5395	00:03:00:00.40	00:02:59;29	−1.370	Time code fell back in 3rd minute
5396	00:01:00:01.40	00:03:00;02	+0.600	Frame numbers 00, 01 skipped
		Minute marks 4-8 not shown		
16183	00:08:59:29.17	00:08:59;29	−0.170	Time code close here
16184	00:09:00:00.20	00:09:00;20	+1.800	Frame numbers 00, 01 skipped
17981	00:09:59:28.97	00:09:59;29	+0.030	Time code almost exact here
17982	00:10:00:00.00	00:10:00;00	0.000	No frame numbers skipped

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

The column headed "Discrepancy" shows the discrepancy between the time code and the theoretical time code, in units of the frame time, to the nearest 0.001 frame.

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 theoretical time code). Thus there is no discrepancy in this season.

Next, we go to the end of the first second. As the frame count reaches 29, the time code and theoretical time code are still together. And at a frame count of 30, the time code clock advances its seconds field, now showing 00:00:01;00. But "by rights", theoretically, it should have advanced the seconds field to "01" 0.030 frame times earlier. And in fact at the instant of interest, when the time code has

just advanced to 00:00:01;00, we see that the "theoretical" time is now already showing 00:00:01:00.03.

And in this way, for each full interval of 30 frames, the time code falls behind the theoretical time code by a further 0.030 frame times.

We next move forward by about one minute. The frame count when the time code clock "would show" exactly 1 minute (except for the skipping of frame counts that will occur there) 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 theoretical time code by 1.770 frames.[12] At 1800 frames, when the time code clock "turns over" to the next second, as we saw before, this constitutes a falling back of the time code time, compared to the theoretical time, by 0.030 frame times.

But at 1800 frames (when the time code clock would, except for frame number skipping, show exactly one minute), the time code also jumps ahead by 2 frames (since frame numbers 00 and 01 are skipped), and thus the time code is now 0.200 frames ahead.

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 [13]. We will first look at things one frame earlier, at a frame count of 3597 frames. We note that the discrepancy here is only −1.570 frames, 0.030 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 that 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 mark."

We see the further progression of this pattern at the "3-minute" point (5396 frames).

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

---

[12] We will see that this is the greatest negative discrepancy (that is, when the time code is behind the theoretical value for that frame).

[13] 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.

Now that we have seen the way the situation progresses, we will shift 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 −0.170 frame times, and the two-frame skip at frame 16183 puts the discrepancy to +1.800 frame times.[14]

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 time has drifted back so it is 0.030 frames ahead of wall time. At 17982 frames (the "10-minute point"), no frame counts are skipped (as this is the beginning of a minute evenly divisible by 10), the time code has suffered its usual 0.030 frame loss compared to theoretical time whenever the seconds field advances, 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 theoretical time code time/wall time and exactly 10 minutes of actual time code time).

For integer frame counts, the greatest negative discrepancy between the time code time and the theoretical time code time is −1.770 frame times, which occurs at the end of the first minute of each 10-minute cycle. The greatest positive discrepancy is +1.800 frame times, which occurs at the "9-minute mark" in each 10-minute cycle.

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

#

[14] This is the greatest positive discrepancy (that is, when the time code is ahead of the theoretical value for that frame).

Appendix C
Algorithm for converting “wall time” to 29.97 DF 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 the customary form) into the corresponding time code under the SMPTE 29.97 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 03 2017.06.27

Author: Douglas A. Kerr

Language: DAK pseudocode

****

// 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

sizeWeeCycle = 1798 // Number of frames between points where frame counts might be skipped.

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

// However, the first interval 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 to the instant of interest

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

frames1 // Number of frames to the instant of interest

frames2 // Number of frames in the "tail" (that is, after all full big cycles have been removed)

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

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

```
numSkips2     // Number of times in 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


// Declare function

HMSF(numFrames)  // Converts argument (in frames) to H:M:S;F form (30 frames =
      1 second)

      // This function not defined here.

// START

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

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

frames2 = frames - (numBigCycles * sizeBigCycle)      // Number of frames in tail
      (beyond all full big cycles)

if (frames2 < (sizeWeeCycle + 2))      // First wee cycle is 2 frames larger than all
      later ones

      numWeeCycles = 1     // If frames2 = 0 this will work out properly

else numWeeCycles = integer ((frames2 - 2) / sizeWeeCycle)  +1     // Number of
      wee cycles (full or partial) if over 1

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

      // No frame counts are skipped in the first wee cycle, which falls at a time
      code minute whose number is

      // an integral multiple of ten.

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

numShips2 = numSkips1 + numSkips2      // Total number of time where frame
      counts are skipped.

framesSkipped = numSkips2 * 2     // Total number of frames skipped

adjustedFrames = frames1 + framesSkipped      // Calculate "adjusted frames" by
      applying all frame skips

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

      // This is the result.

// END
```

#