

SKEW

Introduction

Perhaps it is my own perception, but I feel as though I've seen one dialog in the DVD Lab forum several times. It starts with someone asking a very top level question about randomly playing a clip. It then evolves that they want to the clips to play randomly, but they don't want any clip to repeat until all clips have been played.

This is a contradiction in desires, as will be discussed in the theory section. Briefly, though, the nature of randomness is such that any clip, whether recently played or not, is just as likely to play as any other. What this person wants is to somehow skew (perturbate, change) the random numbers generated so no random number repeats until all random numbers have been selected.

What this algorithm does

This algorithm pseudo-randomly picks clips, but it won't replay any clip until all clips have played. It is presently limited to a maximum of 16 clips. I have another algorithm in progress that will do much the same thing for up to 112 clips, but it is correspondingly more complex to implement. If you can live with 16 or fewer pseudo-random choices, stick with this algorithm; it will be much easier to debug.

I've broken the discussion into pretty much the same areas as before. There's a section on requirements, which tells what you need, a section on implementing the code in your DVD, and a section on the theory behind the algorithm.

Cautionary notes

This is a much more complex (and quirky!) algorithm than those previously presented. It will be harder to get it embedded and running in your DVD layout. Your finished DVD should be tested on several different players, to make sure that each interprets the code correctly.

Skewed Requirements

Just like before, but different

The need for registers just keeps growing! Adding skew requires yet another register to keep track of the clips which have played. The usage is now something like this:

GPRM0 – still used for the pseudo-random number and the seed.

GPRM1 – used for temporary storage

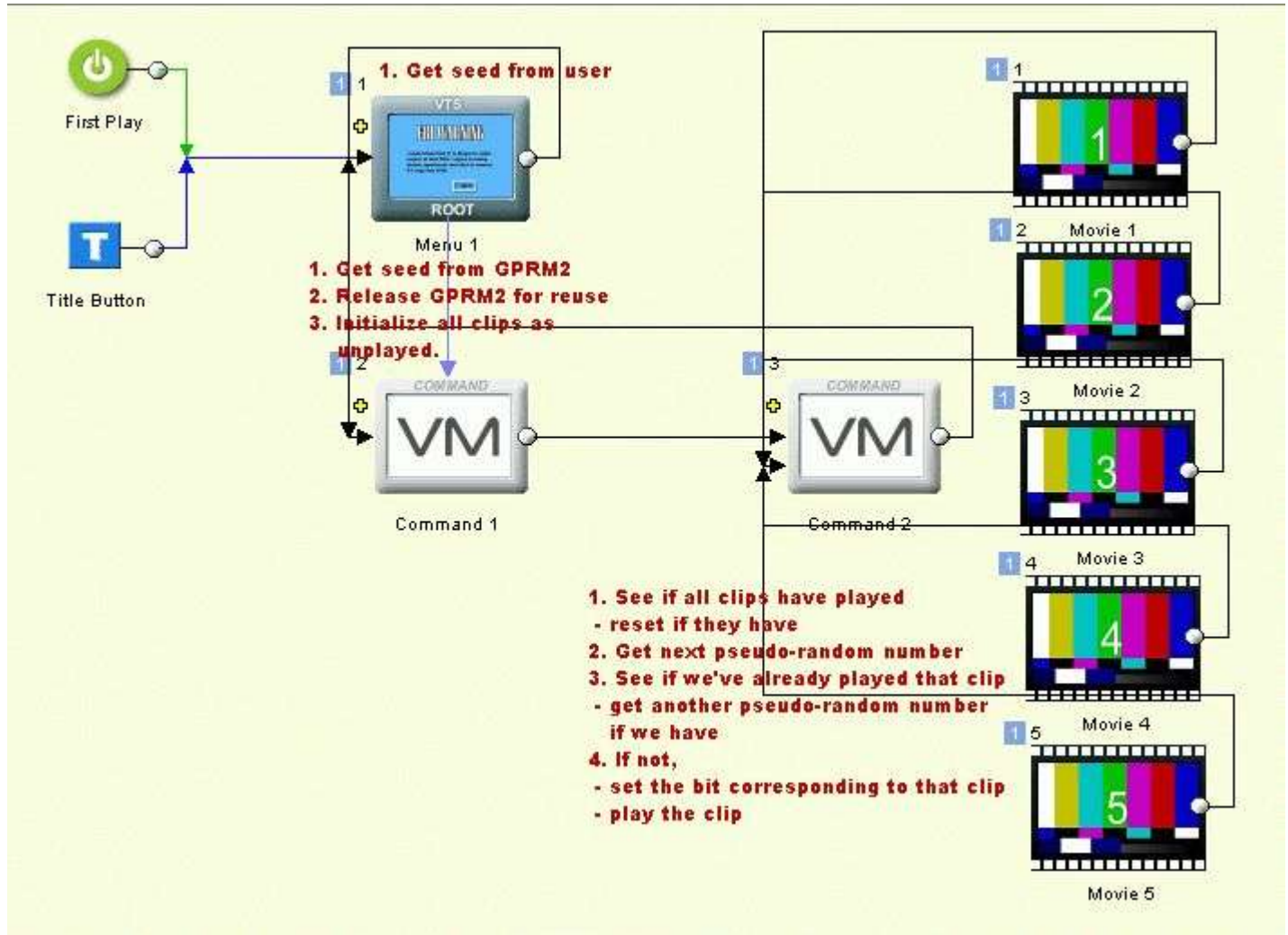
GPRM2 – Initially used as a counter, this becomes the register used to keep track of the clips which have played.

Other than this, the other software and computer requirements remain the same. However, integrating this algorithm has been much more difficult than those previously discussed.

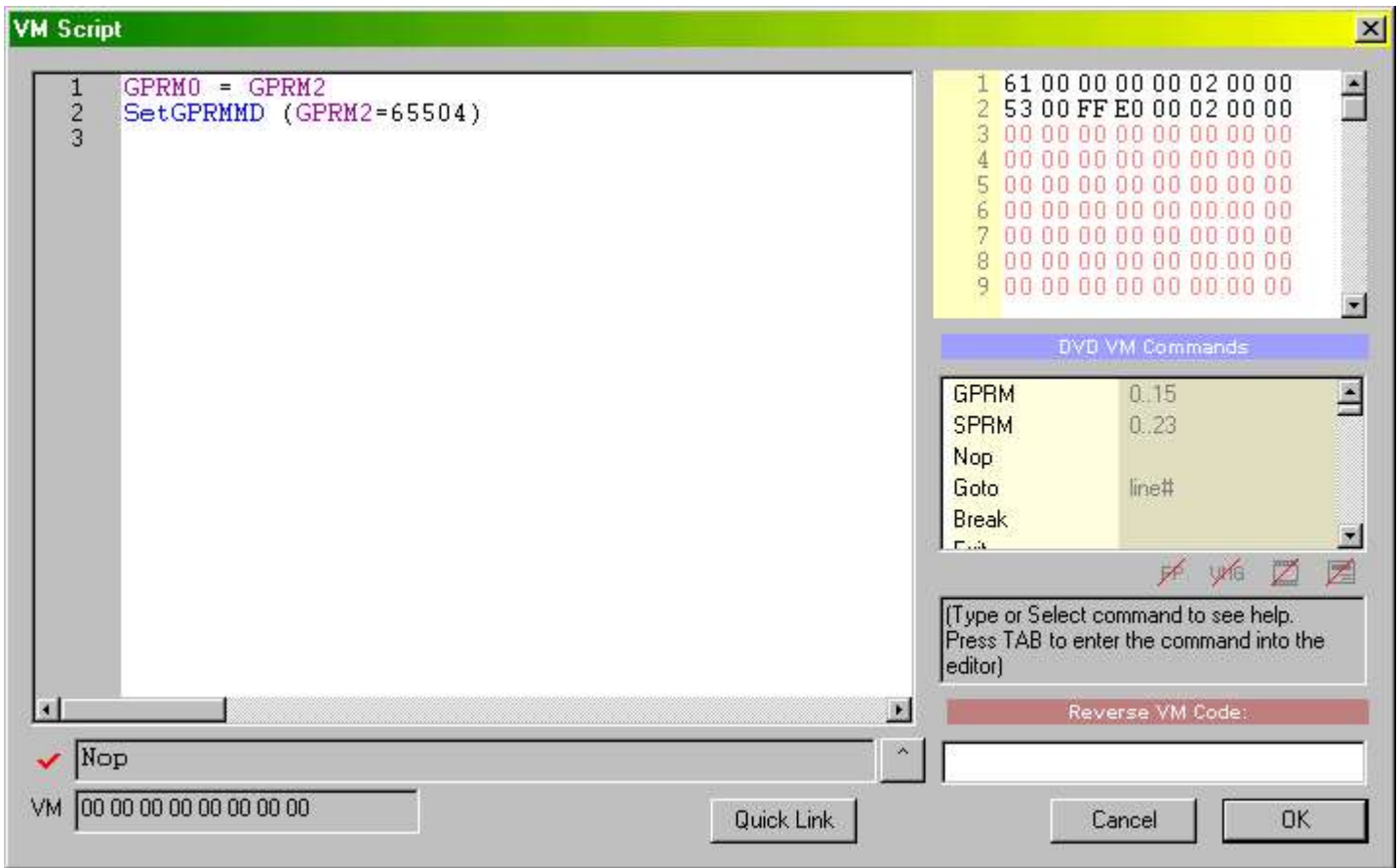
Implementation – Skewed Pseudo-Random Generator

The Code Discussed

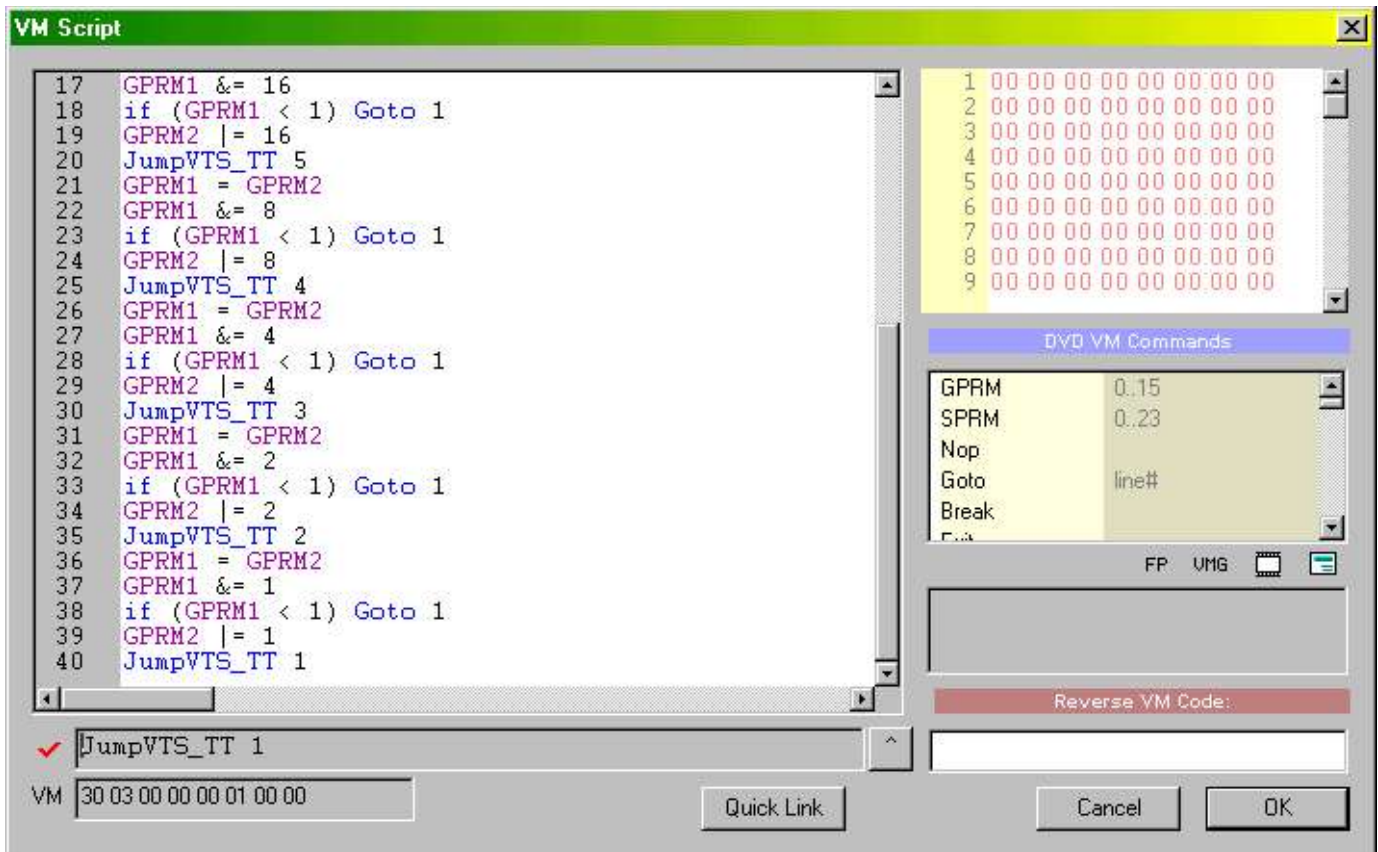
The configuration of the video is identical to that used previously. That is, as it appears below.



As mentioned in the requirements section, there are additional variables associated with the addition of skew. GPRM2 begins its life as a counter, then transitions to holding the record of the clips which have been played. That is, the VM code in command 1 now looks like this:



Prior to entering the generator the first time, GPRM2 is set to 65504, the initial value when no clips have been played, and five clips are to be played. The initial value of GPRM2 for various numbers of clips is shown below. This is also the value to which GPRM2 is reset when all clips have been played. The code in command 2 now looks like this:



Immediately on entering the code shown below, GPRM2 is checked to see if it is bigger than the maximum value we would get if we had played all the clips once. If you have something other than 5 clips, you must replace the “31” in line 1 with the correct value from the table below. You must also correct line 11 to reflect the correct number of clips.

# PGCNs – use in line 11	Line 1 value in command 2	Initial value of GPRM2 in command 1. Reset value of GPRM2 in line 1 of command 2
2	3	65532
3	7	65528
4	15	65520
5	31	65504
6	63	65472
7	127	65408
8	255	65280
9	511	65024
10	1023	64512
11	2047	63488
12	4095	61440
13	8191	57344
14	16383	49152
15	32767	32768
16	65535	0

This points out a limitation of this approach. Once all the clips are played, although the random number generator is still running, our tally of what has been played and what has not starts over. There is a one in five chance (for five clips) that the same clip which just played would play again, and so on. Fortunately, once each clip has played once, it may not matter which appears next.

The next part of the code (lines 2-11) are the same pseudo-random number generator discussed elsewhere. Lines 13-27 steer the ranged pseudo-random number to the correct place. Lines 28 through 91 check to see if the associated clip has just played. If so, they cause the next pseudo-random number to be generated, and it is similarly tested. When a number matches a clip that has not been found, two things will happen:

1. The GPRM2 will be appropriately updated.
2. The routine will transfer control to the appropriate clip.

The logic should keep the code from searching for a clip that has not played but does not exist. It should also reset GPRM2, should it somehow become unbounded. I say “should,” not because I have certain knowledge of a condition in which these things do not happen, but because this code is relatively untested.

Lines 28 through 91 would have benefited from the inclusion of a “GOSUB” and “RETURN” command in the instruction set. Lacking these features, about all one can do is the same inline code, over and over. GPRM2 is being used to hold the record of which clips have played, and which have not. The least significant bit corresponds to clip 1, and the most significant bit corresponds to clip 16. If the bit is set (1), the clip has been played; if the bit is not set (0), the clip has not been played.

1	if (GPRM2 > 31) GPRM2 = 0
2	GPRM0 &= 2047
3	if (GPRM0 > 2046) GPRM0 = 2046
4	if (GPRM0 < 1) GPRM0 = 2046
5	GPRM1 = GPRM0
6	GPRM1 &= 17
7	if (GPRM1 == 1) GPRM0 += 2048
8	if (GPRM1 == 16) GPRM0 += 2048
9	GPRM0 /= 2
10	GPRM1 = GPRM0
11	GPRM1 %= 5
12	if (GPRM1 < 1) Goto 36
13	if (GPRM1 < 2) Goto 31
14	if (GPRM1 < 3) Goto 26
15	if (GPRM1 < 4) Goto 21
16	GPRM1 = GPRM2
17	GPRM1 &= 16
18	if (GPRM1 < 1) Goto 1
19	GPRM2 = 16
20	JumpVTS__TT 5
21	GPRM1 = GPRM2
22	GPRM1 &= 8
23	if (GPRM1 < 1) Goto 1
24	GPRM2 = 8
25	JumpVTS__TT 4
26	GPRM1 = GPRM2
27	GPRM1 &= 4
28	if (GPRM1 < 1) Goto 1
29	GPRM2 = 4
30	JumpVTS__TT 3
31	GPRM1 = GPRM2
32	GPRM1 &= 2
33	if (GPRM1 < 1) Goto 1
34	GPRM2 = 2
35	JumpVTS__TT 2
36	GPRM1 = GPRM2
37	GPRM1 &= 1
38	if (GPRM1 < 1) Goto 1
39	GPRM2 = 1
40	JumpVTS__TT 1

Problems and Limitations

1. Once all desired clips there is no guarantee which clip plays the next time.
2. This technique only works for sixteen clips. The code shown could easily be extended to cover additional clips, simply by using more registers.. I chose fewer registers, feeling that there would not be a frequent need to chose between more than 16 clips. Time will tell.
3. As just alluded, the code gets redundant and messy because of the lack of a GOSUB instruction. I decided to leave the code intact to accommodate 16 clips, and to select the actual number of clips in lines 1 and 11.
4. This is beta code. While the algorithm appears to be sound, and appears to work

correctly on Markham's software player, it has failed to execute correctly on at least one conventional DVD player. I've removed a number of problems which were discovered, but this does not guarantee no other problems exist.

5. If you have more or less clips to play than 5, you must:
 - A. Change line 2 in command 1
 - B. Change line 1 in command 2
 - C. Add or delete comparisons like those in lines 12-15 of command 2
 - D. Add or delete branching code like that in lines 16-20, 21-25, 26-30, 31-35, and 36-40.

1.

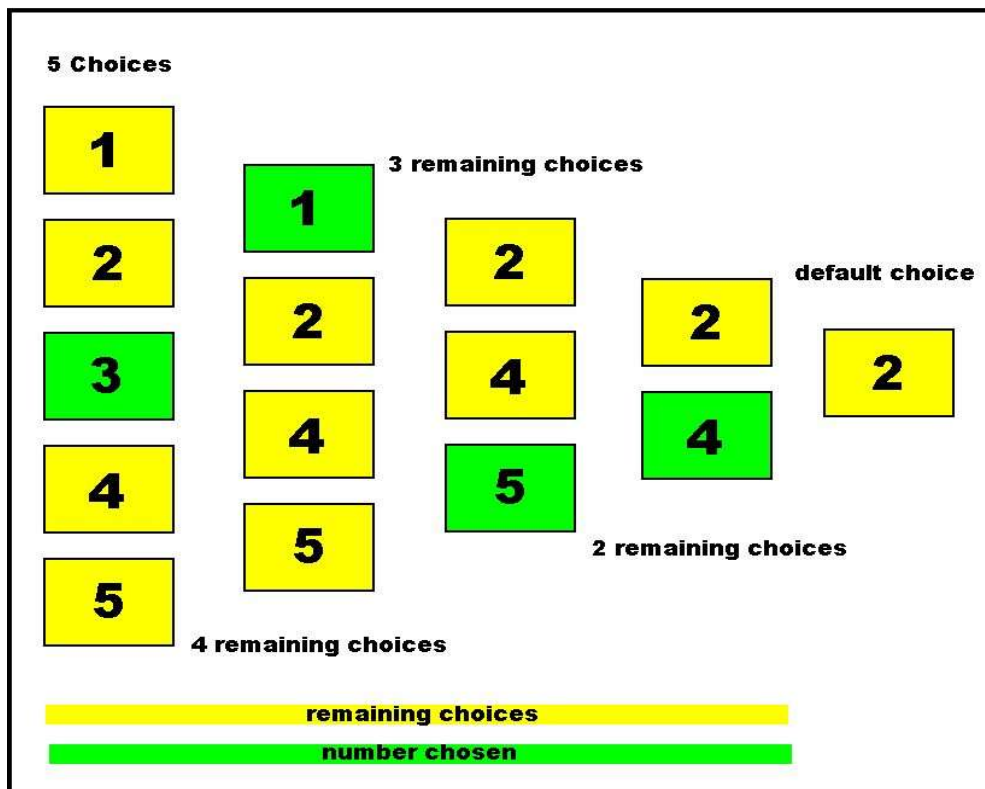
Theory – Skewed Pseudo-Random Generator

Statistics with and without Replacement

In statistics, there are problems in which the terms “with replacement” and “without replacement” are used. What this means is, there is some set of things from which to chose. “Without replacement” means each time some thing is chosen, it is removed from the pool of choices. “With replacement” means it is returned to the pool, and may be chosen again.

In choosing which clip to play, if we want a truly random choice, and don't care if the clip plays again immediately, we are using “with replacement.” If we do not wish a clip to play again until all possible clips play, we want to use “without replacement.” Using the pseudo-random number generator without replacement is the technique discussed elsewhere in these pages.

One of the concepts presented in statistics is that, once selection without replacement is being used, there are only $n!$ Ways in which the selection sequence can occur. That is, if $n=5$ and there are 5 video clips to chose from without replacement, there are only $5*4*3*2$, or 120 ways in which this can occur. This should be more than enough for DVD applications. However, be aware that the possibilities are definitely finite. Strange applications, in which you tie a friend to a chair, and hope to force that person to watch endless variations of the sequence in which clips are played are probably better served by the “with replacement” technique. If you run out of volunteers, I can suggest a few people.



There is an additional limitation on the total number of clips this implementation will support (16) More on that later.

Building in the Skew

If the results are to be at all random, a pseudo-random generator must be at the heart of the approach. Since the one discussed elsewhere seems to fit in the category of “good as any, better than some,” it has been used. The logic we need to implement looks like this:

1. Set up some thing to keep track of what numbers we've already generated.
2. Generate a new pseudo-random number
3. If we've used it in this time through the range, get another pseudo-random number.
4. By default, we haven't
 - A. Keep track of the fact we just used this number
 - B. Branch to the appropriate clip.

The problem is keeping up with what we have recently generated, and what we have not. If there were an infinite supply of registers, the task would be simple: we could just assign a register to each clip, and check to see when the clip last played. Instead, the number of registers needs to be kept at a minimum, which means storing the record of what has played and what hasn't needs to be done in a single register.

Method Used to Track clips Played

I originally had what I thought was a clever way to keep up with this, using a product of prime numbers. I was a little too clever for my own good. It's actually possible to keep up with more clips by just assigning each bit of a register to a clip. This approach is so simple that it hardly needs a theoretical explanation.

But in simple terms, if bit 0 of GPRM2 == 1, clip 1 has played. If == 0, it has not. Similarly, if bit 1 of GPRM2 == 1, clip 2 has played. If == 0, it has not, and so on. GPRM2 is initially set so that all bits except those associated with a clip to be played are set to “1,” and all bits associated with a clip to play are set to “0.”

Each time the pseudo-random number generator is called, it checks to see if all clips have been played (GPRM2 is all “1's”) (line 1 of command 2). If so, GPRM2 is reset. If not, a new pseudo random number is created (lines 2 – 11).

A branch is made to code supporting the candidate correct clip (lines 12-15). A check is then made to see if the associated bit in GPRM2 is set. If so, the clip has been played. If not, the bit is set, and the clip is played.

