RKNanoD (Olive) SHUFFLE 问题说明

20160528

1. **WALKMAN B-183(NANOC)的 shuffle 功能定义如下：**

   当选择播放或者更改 SHUFFLE 模式时，第一首歌曲不算，从下一首歌曲算起：

   例如：当选择第 n 首歌曲播放时，依次会播放如下序列

   n，X,X,X.....

   如果当前歌曲是第 m 首，从非 SHUFFLE 模式换成 SHUFFLE 模式时，依次会播放如下序列：

   m, X,X,X.....

   对应 API 说明如下：

   Int CreateRandomList(unsigned int num, const unsigned int orgSeed0, const unsigned int orgSeed1)

   其中：num: 当前总歌曲数，orgSeed0：初始种子， orgSeed1: 无意义

   unsigned int randomGenerator(DIRECTION_TYPE direction, const unsigned int num)

   direction：方向，num: 当前总歌曲数

   由于内存限制我们使用的线性同余等概率伪随机算法，以下简称算法，算法特点：给定一个初始值，后续序列完全相同（RK 早期 SDK 使用的算法），由于不符合 183 shuffle 的定义，需要更改，客户提出 SHUFFLE 实现方法，但是由于内存的限制，实现不了，最终商讨一个这种的方案，在第一首歌曲不算的情况，我们利用算法来拟合客户的需求，做法如下：

   1) 将算法随机集合扩增到 6 万首以上
   2) 任给一个歌曲种子，产生一个歌曲总个数的子集
   3) 配合上层逻辑，可以基本上满足客户的需求。

   但是只要有耐心在歌曲较少的情况（测试是 5 首），还是能推算出下一首歌曲。

2. **NanoD （Olive）的处理方式**

   NANOD 项目需要解决此问题，让用户推算不出下一首歌曲是什么，因此必须引入新的算法，新的算法需要内存需求，大小为 16K，为支持的最大歌曲数 * 2，由于内存不足，后来采用了一个折中的办法：就是 128 首歌曲以内，消耗 256 字节内存，使用新的算法，128 歌曲以上使用老算法。因为新算法接口定义上不同，初始种子不在有意义，种子是在产生随机序列的时候随机给出的，为了接口的一致性，这里把 128 以上使用的老算法的初始种子给固定为 12 了，也就是说老的算法出来的序列是：

12（初始种子）, 25, 38, 45, 67, 203, 109, 199, 210...

其实这应为 BUG，本来可以在一次修正新算法带来的 BUG 时，可以将其修正，后来考虑新算法稳定性问题，被叫停，所以这个 BUG 一直未被解决。

156	A	#1252 Pause AMS- can't work correctly 1.傳四首歌進去 ABCD 2.shuffle打開 3.先播D的歌曲10秒按暫停 4.一直按FFD退回上一首的順序如下 DCCD DCCD DA bug#歌曲重覆了	EVK2	SVN1533	11/18	Closed		11/16 小馬有復現，應該是王剛修改快切功能造成的 11/17 王剛:與Sony跌規格問題? RK修正shuffle bug? 11/19 Sony找三德討論,提供日文說明報告 11/20 SVN1578已更新

因此大于 128 数据出现序列为

1st time

1）初始种子，25，38，45，67，203，109，199，210,...[此序列是以 12 为初始种子的生成的]

2nd time

2）初始种子，25，38，45，67，203，109，199，210,...[此序列是以 12 为初始种子的生成的]

3rd time

3）初始种子，25，38，45，67，203，109，199，210,...[此序列是以 12 为初始种子的生成的]

问题代码截图：

```
00164: //init global variables which is necessary to random
00165: static void InitGlobeVar(RAND_GENERATOR *pGenerato
00166: {
00167:     pGenerator->seed = Start;
00168:     pGenerator->period = period;
00169:     pGenerator->C = (COEFF_C + 2 * stride) % period;
00170:     memset(gFlagCache, 0, sizeof(gFlagCache));
00171:     SetFlagCache(gFlagCache, num, period, pGenerator
00172: }
00173: //interface to caller
```

```
00433: unsigned int randomGenerator(DIRECTIO
00434: {
00435:     if (rand_seed.total_num <= file_num_bou
00436:     {
00437:         return randomGenerator_128(directio
00438:     }
00439:     else
00440:     {
00441:         if (rand_first_flag == 1 )
00442:         {
00443:             rand_first_flag = 0;
00444:             return server_ori_seed;
00445:         }
00446:
00447:         return randomGenerator_big(directio
00448:     }
00449: }
```

3. 解决方案如下 2 种：

1）使用 WALKMAN B-183(NANOC)算法和规格。

2）将 256 字节扩大到 16K 内存，取消 128 分水岭。