NICAM 软件解码及应用

V 0.1

目录

1 nicam 基本介绍

NICAM 是 near instantaneously companded audio multiplex 准瞬时压扩音频复用，728 表示数据传输速率 728 kb/s. 具体详细介绍参考 ETSI Standard ETS 300 163.pdf 英文标准文档 page7-8。

2 nicam 协议主要部分

2.1 解扰

文档中是先交织，然后是加扰，是编码方式。软件是解码，要先解扰，再解交织。详细说明参见 page 9 4.1.3.
对应的代码是 VOID nicam728_descramble(UINT8* input)。
其中用到一个 pseudo-random sequence。
Initialisation word 初始化及应用的代码对应 UINT8 nicam_prsg_init()， UINT8 nicam_prsg_get(VOID)。
具体原理参见 Page 20.

2.2 解交织

参见 Page9 4.1.2，解交织过程，不详细叙述。
代码如下 VOID nicam728_deinterleave(UINT8* input_data,UINT16* output)；
注意：在调试发现在执行函数前要把相邻字节要相互调换位置，应该是传输是大小端问题
要执行 INT32 swap_data16(UINT8 *data,INT32 len)；

2.3 解扩系数

解扩系数的正确与否直接关系到最后解码数据。
解扩系数获得是通过数据最高六位偶校验得到校验位然后与第 10bit 异或得到解扩系数的一位。详细过程 page13-14.
主要函数为 UINT8 get_Ri(UINT16 data)；
UINT8 parity_check(UINT8 x)；
经过试验发现 mono 和 dual 解扩原理一样，每一帧数据的解扩码一样。
函数 INT32 parse_mono_ra_rb(UINT8 *RA,UINT8* RB,UINT16 *data,UINT16 ctrl)；
Stereo 的解扩系数函数为 INT32 parse_stereo_ra_rb(UINT8 *RA,UINT8* RB,UINT16 *data)；

2.4 解扩

解扩之后的数据就是我们真正需要的数据。详见 Page21.
解扩函数为 UINT16 decode_data(UINT8 range,UINT16 data)；
Mono 和 dual 函数为 INT32 parse_data_mono_data(UINT8 RA,UINT8 RB,INT32 len,UINT16 *data,INT16 *data_out,INT32 count)；
Stereo 函数为 INT32 parse_data_stereo_left_right(UINT8 RA,UINT8 RB,INT32 len,UINT16 *data,INT16 *left_data,INT16 *right_data,INT32 left_count,INT32 right_count)；

3 nicam 解码软件逻辑部分

小节 2 是整个软件解码的基础，软件正确运行都是在这个基础之上。这里先说明一下，每一中断上来的数据都是固定（包含几十帧 nicam 数据），数据的 bytes 数必须 2 幂次方，这是硬件规定的。这里软件在解码是按一帧 91bytes，所以一个中断上来后可定会有剩下的的不够一帧的数据，我们这里就保留位置，在下一个中断上来后，作为这个中断数据。具体代码里都做了处理。

3.1 nicam 数据的确定

判断是否是 nicam 数据主要函数为 INT32 dev_app_read_frames(UINT8* nicam_data,INT32 data_size,UINT8* nicam_data_fix)；

其主要思想为如果连续十六帧帧头为 4e，并且 16 个帧 c0 bit，组合在一起里面有 8 个 0 和 8 个 1，那么正确找到，如果找不到在按位移动数据查找过程也是十六帧作为判断标准，主要函数为 INT32 encorrect_4e_and_ctrl_bits(UINT8 *data,INT32 size,INT32 * pos,INT32* shift)。如果最后还没有解析出来，那边就不是 nicam 数据，或是信号已经很不好，解不出数据。
encorrect_4e_and_ctrl_bits 函数判断正确后，会得到数据需要移动的位数和数据在哪里开始是正确的位置。
按位移动的过程，这里简单叙述。主要函数为 INT32 redecoding_nicam_data(UINT8 *data,INT32 size,INT32 shift)；调用了 UINT8 reparse_data_byte(UINT8 *pdata,INT32 shift)，主要是移动数据得到正确的数据，VOID swap_bytes(UINT8 *p,INT32 size)把最后数据按位从最高位到最低为反过来。

不用每次都这样执行，只用第一次和数据出现错误时才会执行，之后就按照第一次得到信息要移动的位数和记录的位置，直接进行解码。
注意执行完 dev_app_read_frames()后得到的数据都是整数倍的 nicam 帧，这里已经考虑到越界问题。

3.2 确定得到的数据类型

每一次中断都要判断 mon，dual，stereo 是否发生了切换。（这个过程在真实情况下不会有，但是 v56 中支持），我们也做了支持。
主要函数为 UINT8 confirm_mon_dual_stereo(UINT8* data)；做实验发现，信号比较低时，单

独判断一帧就不准了，所以里面做了一个和得到解扩系数一样的算法，概率统计，选取概率最大的那个，保障一定的准确性，经验证得到很好的效果。

3.3 确定左右声道

确定左右通道，防止左右声道串音，确定在 mon 时解出来数据（在 mon 是一帧有声音一帧没有声音）不会放错位置。声道的判断只对 mon 和 dual 有用，stereo 模式是每帧里面都包含左右不用进行判断。

主要的函数 INT32 confirm_left_right(UINT8* input)；通过记录 8 帧 c0 来判断左右声道。只执行一次，之后只要发生错误进行容错处理或是 mon dual stereo 状态发生改变时会再次执行。

3.3 保障声道左右数据量大小一样多

有时候得到的数据是 nicam 帧的奇数倍，这时在 dual 模式再往左右声道 buf 里填数据可能会有一个声道数据会多一帧，为保障整个代码的兼容性，设计为每次吐出数据都是偶数帧，如果是奇数帧那就把最后一帧作为下次数据输出，这里和 3 小节开头部分叙述不是一个地方，需要注意。Mon 和 stereo 本身不存在这样的问题。具体分析参考函数 dev_app_nicam_decode_data 开头和结尾的 switch(mon_dual_stereo)；

4 nicam 软件接口

为方便使用 nicam 解码提供接口函数，再次说明。这里总结一下使用应用调用逻辑，每次在切台或是切通道到 TV 时都会调用 nicam 接口，如果是 nicam，便会执行，如果不是 nicam，就会执行正常过程。如果在 nicam 播放中信号不好时，音频会发消息给应用，应用在切回到正常模式，但是信号好时，现在切换不回 nicam。

4.1 解码用接口

有 VOID dev_app_init_nicam_decode(VOID);用来做初始化使用，初始化一些全局变量的状态。

INT32 dev_app_nicam_decode_data(UINT8 *data,UINT32 size,INT16 *out_left,INT16 *out_right);
此函数主要用来进行 nicam 解码，解码后供后处理用。

4.2 应用调用的接口

首先，为 VOID serv_app_api_get_nicam(INT32 *param)，主要是用来判断是否为 nicam。

其次，serv_app_api_get_sound_mode，最后会调用到 INT32 dev_app_nicam_get_mon_dual_stereo(VOID);能够判断是否是 mon 或是 dual 还是 stereo。

4.3 发送消息

当信号不好时，nicam 数据就会解析不出来。这是就要切换到正常模式，这一个功能的实现是靠发消息给应用。可以参考如下：

```
if(nicam_error_count++==16) {
COS_SendRmtEvent(chip_shared_get_mmi_task_handle(),
EV_FORMAT_NICAM_DECODE_ERROR,0, 0, COS_EVENT_PRI_NORMAL);
}
```

此功能过于简单，请读者自行了解。

5 重点和注意的地方

Nicam 解码应该是硬件解，但是硬件本身不支持，特此采用软件解码。需要有些注意的地方，为读者指明。

5.1 重点

主要是 3.1 节和第 2 节，基本构成了解码的主要部分，如果读者不是很了解，可以先按照第 2 节把程序走一步，运行调试。再次基础上，在调节整个软件逻辑。这个解码过程其实内容不多，但是很复杂。

5.2 注意的地方

需要注意的地方很多。首先，在调试移植到板子的时候，发现根本跑步起来，一直挂掉，取指令错误。后来，经过与做 cpu 的人了解，我司的 cpu 不支持奇地址访问，只要访问必挂，只能在软件上做修改，很多地方先放到临时变量里，如 nicam728_deinterleave、shift_to_l、reparse_data_byte 等。

其次，做实验发现信号还算很好的时候，也会出错。这里增加了概率统计的方法，能够有效避免。如 confirm_mon_dual_stereo、parse_stereo_ra_rb、parse_mono_ra_rb。

再次，解码过程中增加了容错处理，保障解码软件具有一定的在错误状态恢复状态。但是在错误过多时，相信客户都无法忍受，只好把 nicam 模式自动切换到正常模式。这里的切换时实在四个中断数据（一个中断有 40 多帧 nicam）解析不出来时。

还有，需要注意的是，一直执行找 nicam 头的处理函数，bcpu 会挂掉，这个已经加到容错处理里。这里列出来，希望读者明白，一直执行找头出来，会拉长 bcpu 处理时间，这时中断会不断发消息上来，把 bcpu 消息队列撑爆。这里已经做了处理。

最后，提供的接口用到的两个 cpu 通信，当 xcpu 读取 bcpu 时，cpu 首先会访问 cache 里的数据，这就有问题了 xcpu 可能会得到一个无效的数据。在这里采用的是刷 cache 的方法。其函数为 dev_cache_clean_dcache_range(param, 4);

6 总结

以上把整个解码及其调用叙述一边，希望对读者有所帮助。要想深入了解及其应用还要自己动手调试和看文档。整个代码基本都是本人自己完成，有些逻辑和不对的地方，请指出说明。邮箱 guanjiewu@rdamicro.com.