AN13793

基于 NXP RT685 FLAC 移植及编解码性能评估

Rev. — 30 November 2022 Application note

Document information

Information	Content
Keywords	FLAC
Abstract	本文档介绍了 FLAC 编解码库在 i.MX RT6xx 上的移植，对比了 CM33 和 HiFi4 DSP两核性能差异，最后使用 GNU profiler 工具分析FLAC库中耗时的函数，为进一步的性能优化提供了建议。

1 介绍

本文档：

- 介绍 Free Lossless Audio Codec (FLAC) 编解码库在 i.MX RT6xx 上的移植。
- 对比 CM33 和 HiFi4 DSP 两核性能差异。
- 使用 GNU profiler 工具分析 FLAC 库中耗时的函数。
- 为进一步的性能优化提供了建议。

2 FLAC 简介

FLAC全称 Free Lossless Audio Codec，即无损的音频编解码器。与我们熟知的音频压缩格式 mp3 类似，但 FLAC 的优点是无损压缩，即音频通过 FLAC 编码后不仅能够减小文件大小，而且还能够完全解码还原音频。FLAC 编码主要分为以下 4 个步骤：

1. 分块
2. 通道间去相关
3. 多项式拟合或线性预测
4. 残差编码（Rice 编码）

更多详细信息和版本差异详见 FLAC。以下评估均以 LibFLAC1.3.4 版本作为示例。

3 测试环境

3.1 i.MX RT685 介绍

- 新一代 Cortex-M33 控制处理器内核，运行频率高达 300 MHz。
- Cortex-M33 包含两个协处理器：
  - PowerQuad 硬件加速器用于（固定和浮点运算单元）DSP 功能。
  - 提供 CASPER Crypto 协处理器，对某些不对称加密算法所需的各种函数进行硬件加速。
- 高度优化的 Cadence Tensilica HiFi 4 DSP 处理内核，运行频率高达 600 MHz。
  - 硬件浮点运算单元每周期多达 4 个单精度 IEEE 浮点运算 MAC。
- 支持 8 个区域的 Cortex-M33 内置存储器保护单元（MPU）。
- 高达 4.5 MB 系统 SRAM，CPU 和所有 DMA 引擎均可对其进行访问。
- 提供丰富的外设、EdgeLock 400A 安全，功耗极低，支持低功耗模式和快速唤醒功能。

3.2 i.MX RT600 EVK 介绍

硬件评估板使用 i.MX RT600 EVK（MIMXRT685-EVK REV E2），其采用恩智浦 Arm Cortex-M33 高级内核，并结合了高度优化的 Cadence Tensilica HiFi 4 DSP 处理器内核。适用于机器学习/AI、语音及视频应用。满足我们对 FLAC 的评估要求。

Figure 1. 硬件平台

4 FLAC 移植

从 GitHub 上我们可以下载到最新版的 FLAC，FLAC 库包含了压缩（encoder）和解压（decoder）两部分的源码。我们这里以 1.3.4 版本为例。

4.1 FLAC 库目录结构

FLAC 库包含 C 和 CPP 两个版本。由于版本的迭代以及方便用户在不同硬件平台、编译器上评估，FLAC 是越来越大。除了核心的编解码源码外，还包括各种说明文档，测试代码，测试、编译脚本等。对于我们嵌入式设备来说，只需要基础的编解码即可。所以移植过程中仅如Figure 2中的三个目录比较重要，其余目录均可忽略：

1. **Src/libFLAC** 目录：核心代码位置，我们使用 C 版本的源码放在 libFLAC 目录。该目录主要分为源码和 include 头文件两部分。FLAC 官方将一些耗时的函数（如 fixed, lpc, stream_encoder），在一些通用的硬件平台（如sse2，ssse3，neon，vsx等）做了对应指令集的优化。如 fixed_intrin_sse2.c，表示对 fixed.c 函数针对 sse2 指令集（Intel平台支持）进行了优化。由于我们使用的CM33和HiFi4并没有支持，所以我们使用基础的版本即可。
2. **Examples/c** 目录：演示代码位置，也分 C 和 CPP 两个版本，我们依旧使用 C 版本。目录下分别有 encode 和 decode 的 main.c，分别代表压缩和解压两个 Demo。这是一个很好的 libFLAC API 使用的参考例程，对于移植有很大的帮助。
3. **Include** 目录：除了 libFLAC/include 目录外，还有部分头文件位于该目录下的 FLAC 和 share 子目录，同样需要包含进去。

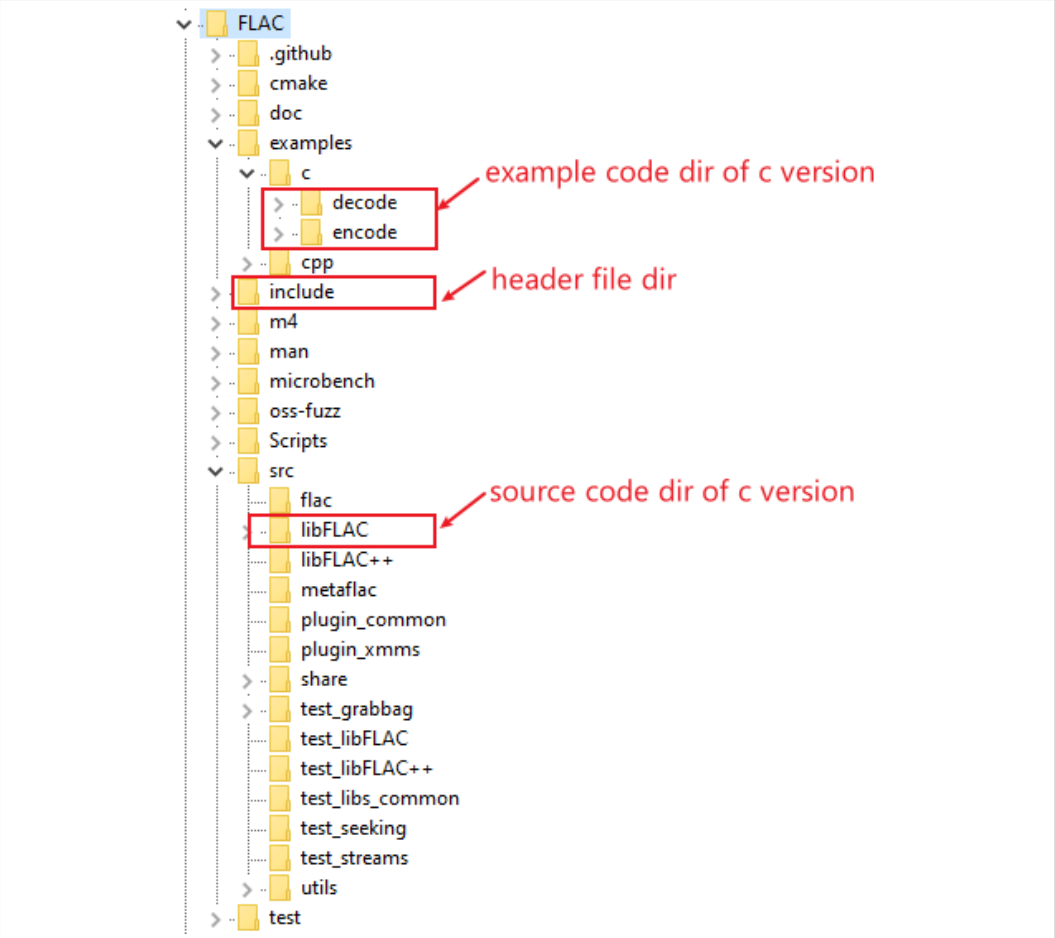

Figure 2. FLAC 源码目录结构

Table 1 列出了 FLAC Decode 和 Encode 移植需要依赖的文件清单。

Table 1. FLAC 核心源码清单

序号	函数名	函数功能	移植需要的更改	**Decode** 需要依赖的文件	**Encode** 需要依赖的文件
1	Bitmath.c	提供 silog2 计算函数		√	√
2	Bitreader.c	读 Bit 流接口		√	√
3	Bitwriter.c	写 Bit 流接口			√
4	Cpu.c	提供查询 CPU 版本型号的接口函数			√
5	Crc.c	CRC 校验接口函数		√	√
6	Fixed.c	固定阶数多项式拟合接口函数		√	√
7	Format.c	FLAC 格式化相关函数		√	√
8	Lpc.c	线性预测计算相关接口函数		√	√
9	Md5.c	提供 MD5 校验接口函数		√	√
10	Memory.c	动态内存申请相关函数		√	√
11	Metadata_object.c	生成 FLAC metadata 的接口函数			√

Table 1. FLAC 核心源码清单*...continued*

序号	函数名	函数功能	移植需要的更改	Decode 需要依赖的文件	Encode 需要依赖的文件
12	Stream_decoder.c	解码核心函数	√	√	√
13	Stream_encoder.c	编码核心函数	√		√
14	Stream_encoder_framing.c	编码组成帧函数			√
15	Window.c	窗函数函数，haming，hann，tukey			√

可以看出移植需要修改的就是 Stream_decoder.c 和 Stream_encoder.c 两个文件。也可以发现相较于 Encode 和 Decode 依赖的文件更少。

4.2 使用 FatFs 替代 C 库文件系统

通过查看 FLAC 源码我们可以发现，example/c 例程是基于 fopen，fwrite，fread 等标准 C 库函数接口读写文件的方式去演示 encode 和 decode 的过程。两个例程的主要流程如下：

1. 在 encode 例程中：
   a. libFLAC 读取输入的 WAV 音频文件，根据 WAV 音频头 44 byte，去设置 encoder 参数。
   b. 其余数据为待压缩音频数据循环传递给 encoder 即可。
   c. 回调输出压缩后的音频文件。
2. 在 decode 例程中：
   a. libFLAC 读取 FLAC 文件的 bit 流，通过同步标志找到每个 frame。
   b. 通过存储在每个 frame 头的压缩方法标志，找到对应解压方法。
   c. 解压输出 WAV 文件。

FLAC 的编解码都有两套函数接口：一个是基于文件读写的（Figure 3 中的绿色箭头），一个是基于数据流的（Figure 3 中的橙色箭头）。由于我们需要获取最终的编解码文件，所以采用基于文件的方式去移植。那么我们移植主要工作就是：将 FLAC 库的文件系统改为我们嵌入式系统支持的 FatFs 文件系统，如 fopen 改为 f_open，fwrite 改为 f_write。框图如Figure 3所示。

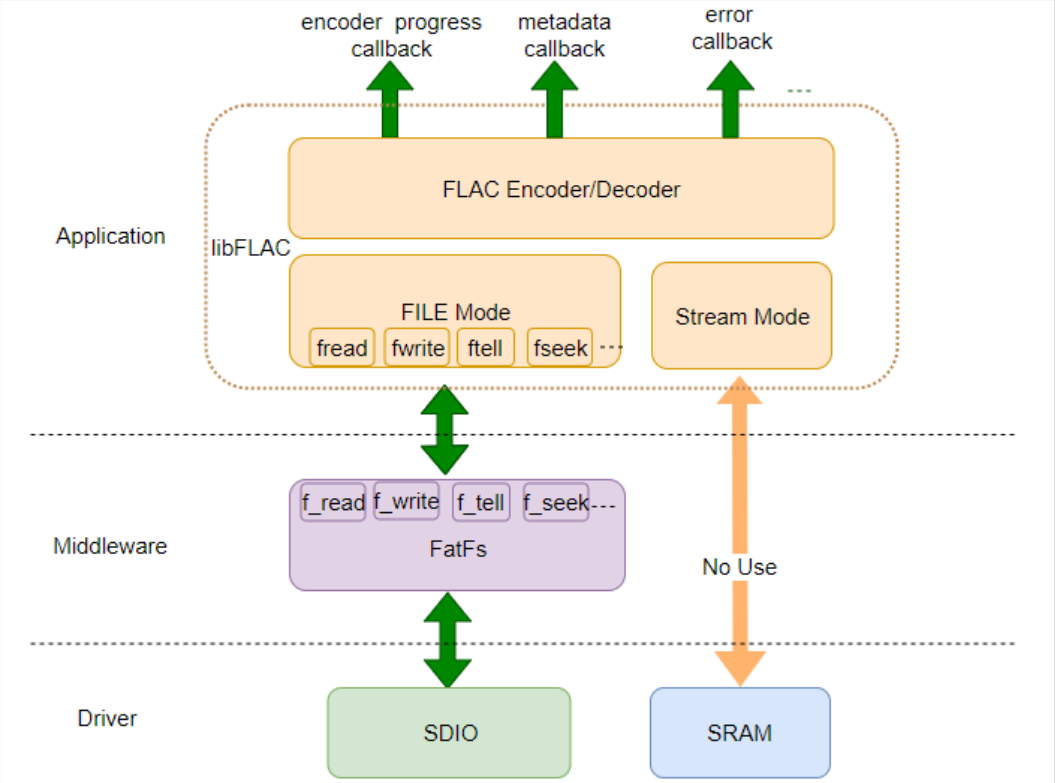

Figure 3. FLAC 加入 FatFs 后数据流图

4.3 添加 SD Card 驱动

对于 SDCARD 驱动，硬件使用 RT6xx 的 SDIO 硬件接口，4bit 传输模式。为了方便 SD 卡可以在 PC 上面读写，所以我们使用 FatFs 开源文件系统。对于 FatFs 的移植，我们只需要在 diskio.c 实现基础的 SD 卡读写扇区，获取寄存器等接口，就可以操作 SD 卡的文件系统了。具体代码可以参考 SDK 的 demo：i.MX_RT600\SDK\boards\evkmimxrt685\sdmmc_examples\sdcard_fatfs。

4.4 添加宏定义配置

FLAC 源码中有许多的宏定义，可以很方便的帮助我们切换到不同平台、开启/关闭某些功能等。Table 2 列出了在移植中我们需要添加的宏定义。

Table 2. 需要添加的宏定义表

需要添加的宏定义	说明
FLAC__HAS_OGG=0	我们测试使用 Native FLAC，不使用 Ogg FLAC，所以这里将 OGG 关闭。
PACKAGE_ VERSION="1.3.4"	手动定义 package version 字符串。
HAVE_LROUND	表示 IAR 和 Xplorer 支持 lround 函数，不使用 build_in 实现。
HAVE_STDINT_H	表示 IAR 和 Xplorer 支持 stdint.h 头文件。
FLAC_EMBEDDED	FLAC 使用移植后的嵌入式接口。

4.5 增加堆栈区大小

另外需要特别注意的是，libFLAC 中用到了很多动态内存申请（调用系统 malloc，free 函数），所以我们的工程的 heap 需要设置的足够大，不然 libFALC 会初始化失败。对于 stack 栈区大小，设置为 8 kB 即可。heap 区需要 430 kB（encode）或 100 kB（decode）。具体大小细节可参考 Section 6.3。

4.6 移植到 DSP 差异

由于只在 CM33 上做正确性验证，所以 DSP 不用移植 FatFs 文件系统，对于上述 FatFs 相关的都改为读写 SRAM 内存即可。关于函数接口设计，参考 Section 5.5。

5 性能评估方法

为了比较全面的评估 FLAC 库在 RT685 上的性能表现，我们首先任选一个音频对移植的正确性做出判断。如果证明移植成功，再使用不同的音频文件在 CM33 和 HiFi4 两个核上分别进行压缩和解压，对比性能差异。

Note:

性能差异主要指两核各自在最大频率下，运行 FLAC 耗时的差异。

5.1 移植正确性验证

由于 FLAC example 默认使用读写文件的方式，而且保存为文件后很方便在 PC 上验证，所以我们首先在 CM33 核验证移植正确性。简单来说就是先压缩，再解压，最后将解压后的文件与原始文件对比。具体步骤如下：

1. 将待测试的音频文件 raw.wav 放入 SD 卡。
2. 使用 FLAC 预设的 0-8 这 9 个压缩等级（0 表示压缩率最低即压缩后文件最大，8 为压缩率最高即压缩后文件最小，详见 Table 4）。分别对 raw.wav 进行压缩，压缩后文件命名为 level_0.flac,...level_8.flac，也写入 SD 卡中保存.
3. 循环将 level_0.flac...level_8.flac 使用FLAC库解压为 level_0.wav...level_8.wav 这 9 个文件，也存储到 SD 卡的 FatFs 文件系统中。
4. 将 SD 卡插入 PC，使用 Beyond Compare 软件挨个对比 raw.wav 和 level_0.wav... level_8.wav 这 9 个文件，其与原始音频完全一致则证明移植成功。

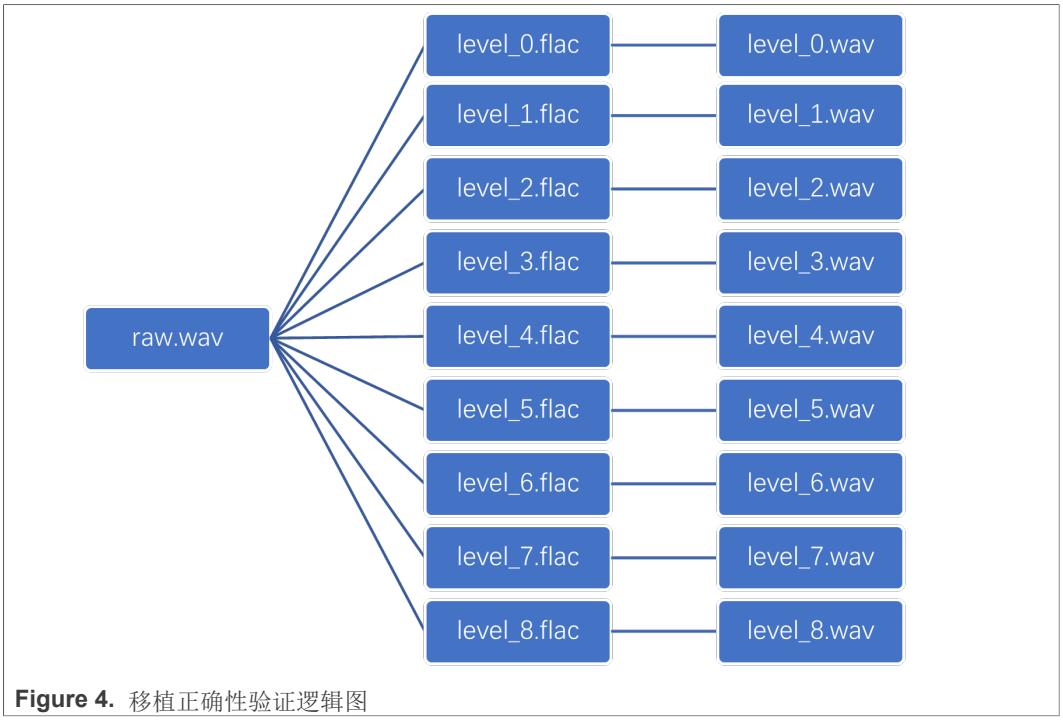

Figure 4. 移植正确性验证逻辑图

5.2 测试音源的选择

为了充分验证和评估FLAC在 RT685 上的性能，我们使用3类音频源作为测试音频：

1. 正弦波扫波
2. 英文对话
3. 音乐

同时还有采样率和声道的差别，测试音频具体参数如下表。

Table 3. 测试音频表

No.	Raw file name	Signal	SR	CH	Resolution	Duration(s)	File size (kB)	
1	16 kHz_mono_ sine_10s.wav	Sine wave	16 kHz	mono	16 bit	10	313	正弦扫波信号（Sine wave）。由于测试音频有16 KHz和 48 KHz，根据采样定律且使频率保持一致，所以使用 0 - 8 KHz 线性增加的正弦信号。
2	48 kHz_mono_ sine_10s.wav	Sine wave	48 kHz	mono	16 bit	10	938	
3	16 kHz_mono_ speak_10s.wav	Speak	16 kHz	Mono	16 bit	10	313	一段英文对话（Speech）。No.4 和 No.6 双通道音频的左右声道数据完全一致。
4	16 kHz_stereo_ speak_10s.wav	Speak	16 kHz	Stereo	16 bit	10	626	
5	48 kHz_mono_ speak_10s.wav	Speak	48 kHz	Mono	16 bit	10	938	
6	48 kHz_stereo_ speak_10s.wav	Speak	48 kHz	Stereo	16 bit	10	1876	

Table 3. 测试音频表...*continued*

No.	Raw file name	Signal	SR	CH	Resolution	Duration(s)	File size (kB)	
7	48 kHz_mono_ music_10s.wav	Music	48 kHz	Mono	16 bit	10	940	一段中文歌曲（Music）。No.8 代表双通道数据类似但不完全一样的数据，No.9 表示两通道完全不同的音频（左声道 Music，右声道 Sine wave）。
8	48 kHz_stereo_ music_10s.wav	Music	48 kHz	Stereo	16 bit	10	1878	
9	48 kHz_stereo_ music_sine_10s.wav	Music + Sine	48 kHz	Stereo	16 bit	10	1878	

5.3 评估压缩性能思路

为了评估压缩性能，请执行以下操作：

1. 使用 Table 3 所示的 9 个测试音频放入 SD 卡中。
2. 在 RT685 中依次读取这 9 个文件，然后再依次使用 9 种不同的压缩等级去压缩，压缩后文件分别命名为 myEncode0.flac，myEncode1.flac … myEncode8.flac。
3. 将 SD 卡插入 PC，验证文件能否播放，同时记录文件大小和压缩耗时。

libFLAC 预先定义的 9 种压缩等级如 Table 4 所示，主要参数含义：

- compress level：FLAC 预设压缩等级 0-8。
- do_mid_side_stereo：选择是否在“通道间去耦合阶段（Interchannel Decorrelation）”阶段，将 stereo 音频做 mid_side 处理。
  - 如果为 TRUE：将使用 mid_side 模式编码，即使用 mid 和 side 数据替代原始左右路数据，目的是粗略的降低数据冗余。其中 mid 含义为 (left channel-right channel)/2，side 含义为 left channel – right channel。
  - 如果为 FLASE：将使用 Independent 模式编码，即左右通道分别独自编码。
- loose_mid_side_stereo：该选项表示“松”的 mid_side 模式。
  - 如果为 TRUE：Encoder 会在 Independent 和 mid_side 之间自适应的选择最优解（do_mid_side_stereo 必须为 TRUE 才能起作用）。
  - 如果为 FLASE：Encoder 不会自适应选择，即完全依赖于 do_mid_side_stereo 的选择。
- max_lpc_order：最大线性预测编码 lpc 的阶数，阶数越大得到的余差（residual）就越小，但也越耗时。阶数默认范围 0-32。
- max_residual_partition_order：余差分区最大的阶数。如果为 0 则不分区，整个 blocksize 的 residual 都使用同一个rice_parameter 去编码。如果不为零，这平均分为 2^max_residual_partition_order 个分区，每个分区使用各自的 rice_parameter 去编码，这样由于需要多计算 rice_parameter 而变得更耗时，但是最后 rice 编码后的文件会更小。
- *apodization：语音预处理阶段加窗函数的参数选择（加窗用于降低分帧后的截断效应）：
  - Tukey（P）窗：0 <= P <= 1， P 为参数，指定了窗函数的锥度。如果 P = 0，对应为矩形窗，如果 P = 1，对应为汉宁hann 窗。注意 P 为科学记数法，5e-1 = 0.5。
  - subdivide_tukey 窗：是 partial_tukey 和 punchout_tukey 的一种更有效的重新实现，可以回收尽可能多的数据。

  注意：固定多项式预测默认帧大小（blocksize）是 1152 个采样点，LPC 默认 blocksize 是 4096 个采样点。

更多选项请参考官网手册。

Table 4. FLAC 库预先定义的压缩等级

Compress level	do_mid_side_stereo	loose_mid_side_stereo	max_lpc_order	qlp_coeff_precision	do_qlp_coeff_prec_search	do_escape_coding	do_exhaustive_model_search	min_residual_partition_order	max_residual_partition_order	rice_parameter_search_dist	*apodization	Block size
0	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	0	3	0	tukey (5e-1)	1152
1	TRUE	TRUE	0	0	FALSE	FALSE	FALSE	0	3	0	tukey (5e-1)	1152
2	TRUE	FALSE	0	0	FALSE	FALSE	FALSE	0	3	0	tukey (5e-1)	1152
3	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	0	4	0	tukey (5e-1)	4096
4	TRUE	TRUE	8	0	FALSE	FALSE	FALSE	0	4	0	tukey (5e-1)	4096
5	TRUE	FALSE	8	0	FALSE	FALSE	FALSE	0	5	0	tukey (5e-1)	4096
6	TRUE	FALSE	8	0	FALSE	FALSE	FALSE	0	6	0	subdivide_tukey (2)	4096
7	TRUE	FALSE	12	0	FALSE	FALSE	FALSE	0	6	0	subdivide_tukey (2)	4096
8	TRUE	FALSE	12	0	FALSE	FALSE	FALSE	0	6	0	subdivide_tukey (3)	4096

5.4 评估解压性能思路

为了评估解压性能，请执行以下操作：

1. 将Section 5.3压缩后的 myEncode0.flac，myEncode1.flac…… myEncode8.flac 这 9 个 FLAC 文件，分别进行解码操作，解码后命名为 mydecode0.wav，mydecode1.wav…… mydecode8.wav。
2. 在 PC 上将这 9 个 wav 文件与Section 5.3对应的源文件 test.wav 使用 beyond compare 的 HEX 方式去对比，文件完全一致则证明 FLAC 的无损压缩是有效的。同时记录解压耗时。

5.5 去除读写文件耗时影响

由于以上 FLAC 的编解码测试都需要读写 SD 卡文件，总所周知，MCU 读写 SD 卡非常慢，可能会影响最终的测试结果。所以我们Section 5.3和Section 5.4验证通过后的基础上（即编解码功能验证通过），将整个音频文件预先读到共享的 SRAM 中（内存分配查看Section 5.6），且压缩/解压后的文件也直接写入 SRAM（这里不再验证文件的正确性，只评估耗时），由此来评估单纯 FLAC 编解码耗时。

音频文件放到了 SRAM 区域，所以设计了一套操作 SRAM 读写的接口去替换之前 FatFs 的 f_write，f_read，f_tell，f_size等接口。如下：

```
void virtual_fs_api_resetReadPr(void);
uint8_t virtual_fs_api_readbuf(uint8_t* data, uint32_t len);
```

```
uint8_t virtual_fs_api_copyMusic2Sram(char *inputfile);
uint8_t virtual_fs_api_readbuf_autoMovePr(uint8_t *data,
 uint32_t len);
uint8_t virtual_fs_api_seek(uint32_t absolute_byte_offset);
uint32_t virtual_fs_api_tell(void);
uint32_t virtual_fs_api_size(void);
bool virtual_fs_api_eof(void);
```

5.6 时间统计方法

时间统计的准确性直接影响到了最终的结果，所以定时器的选择格外重要。我们这里使用 RT6xx 的 OS Event Timer，它的优点是 64 bit 的格雷码计数器，可以不用考虑溢出需要处理中断累加计数的问题。而且 CM33 和 HiFi DSP 均能直接访问，避免了两核使用不同定时器带来的误差。时钟源选择 Cortex-M33 clock 即 HCLK系统的时钟（这里是250 MHz）。

CM33 和 HiFi4 的代码基本一致，需要注意的是 CM33 是使用 OSTIMER0，HiFi4 DSP 使用 OSTIMER1。核心代码如下：

```
static uint64_t gs_encodeStartTimestamp;  // starttimeStamp
static uint64_t gs_encodeEndTimestamp;    //  endtimeStamp

CLOCK_AttachClk(kHCLK_to_OSTIMER_CLK); // use HCLK 250MHz clc
// Switch clc source need reset os event timer (recount from
 zero).
*(uint32_t *)(0x40020000 + 0x40) = 0x01 << 27;  // PSCCTL0_SET
*(uint32_t *)(0x40020000 + 0x70) = 0x01 << 27;  //
 PSCCTL0_CLROSTIMER_Init(OSTIMER0);
...
gs_encodeStartTimestamp =
 OSTIMER_GetCurrentTimerValue(OSTIMER0);
EncodeDemo(inputfilename, outputfilename, iter);
gs_encodeEndTimestamp = OSTIMER_GetCurrentTimerValue(OSTIMER0);
PRINTF ("Timer consume:%.3fms\n", (gs_encodeEndTimestamp-
gs_encodeStartTimestamp)/250000.0);
```

5.7 内存区域划分

i.MX RT6xx 拥有 4.5 MB 的大内存，满足将整个音频文件放入 SRAM 的条件。这里将存储在 SD 卡内待压缩/解压的文件预先通过 CM33 核的 SDIO 接口经 FatFs 存储到共享的 Music 区域，该区域位于 4.5 MB SRAM 的最高地址。将 DSP 和 CM33 的代码，data 放到 SRAM 的其他区域。

对于 CM33 上的 FLAC 测试来说，我们只需要运行一个 CM33 代码就可以进行测试。CM33 可独占除了 Music 外的 SRAM 区域。对于 0x0000 0000~0x0007 FFFF 是 SDK 默认的共享保留内存，这里我们也不作修改。最终 CM33 运行的内存划分如Figure 5所示。

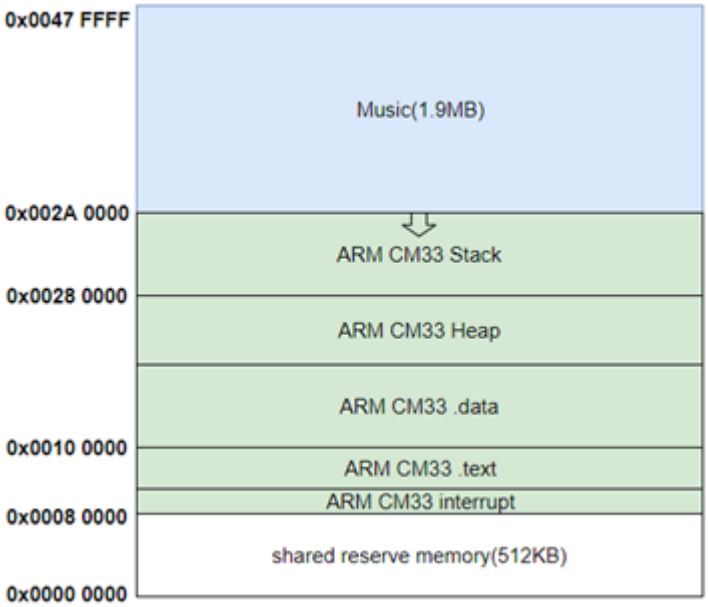

Figure 5. CM33 运行编解码内存划分图

对于 HiFi4 DSP 上的 FLAC 测试来说，由于 HiFi4 DSP 不能单独启动，需要依赖 CM33 启动 HiFi DSP 必要的时钟，初始化等，所以实际需要运行两套代码。CM33 还负责将音频文件读取到 Music 区域，但对比 HiFi4 需要执行的 FLAC 编解码来说 code 和 data 都更少。所以 CM33 占用空间远小于 HiFi4，最终内存划分如Figure 6所示：

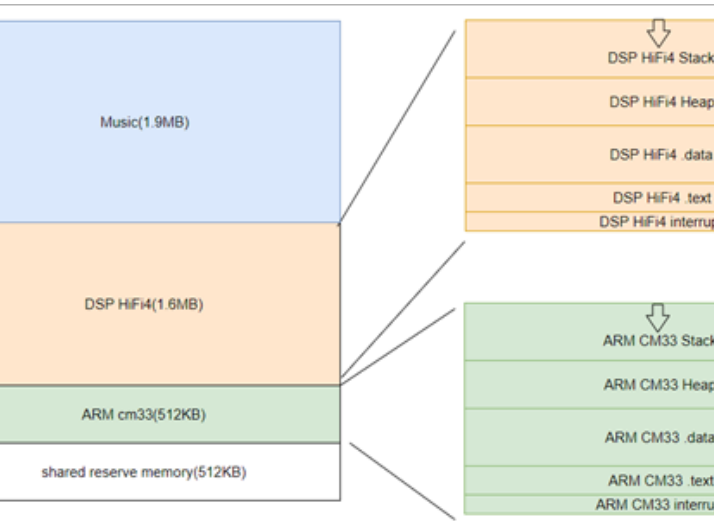

Figure 6. DSP 运行编解码内存划分图

6 测试结果

本章测试结果均基于如 Table 5 的测试环境。

Table 5. 测试环境表

项目	CM33	DSP
运行主频	300 MHz	600MHz

Table 5. 测试环境表...*continued*

项目	CM33	DSP
代码执行位置	SRAM on Chip	SRAM on Chip
IDE 版本	IAR for Arm 9.20.4	Xtensa Xplorer Version 9.0.18
SDK 版本	SDK_2.12.0	DSP Configuration V8.0.15
IDE 优化等级	-Ofast	-O3
其它编译选项	Library -Full -FPU	-mcoproc -LNO:simd

Note:

这里之所以选择 IAR 作为 CM33 核的 IDE，是因为经过测试后发现：IAR 与 MCUXpresso 和 Keil 在相同代码情况下对比，它的运行速度是最快的。

6.1 压缩表现

对于Section 5.2中的 9 个文件，我们分别在 CM33 和 HiFi4 两个核上进行压缩，可得到如下 9 个表。每个表除去 Title 行外由 9 行构成，每行代表一个压缩等级。每一列含义依次是：

1. **FLAC** 压缩等级：范围 0-8，FLAC 预先定义的压缩等级（详见Table 4）。
2. 原始文件大小（单位 kB）：这里是待压缩原始 WAV 文件的大小。
3. 压缩后文件大小（单位 kB）：这里是压缩后FLAC文件的大小。
4. 压缩率：压缩后文件大小/原始文件大小 * 100 %。
5. **CM33** 核压缩耗时（FLAC 文件写入 **SD** 卡的版本，单位秒 s）：CM33 核从开始压缩到压缩完成的时间（此版本仅用于获取压缩后的文件来验证压缩正确性）。
6. **CM33** 核压缩耗时（FLAC 文件直接写入 **SRAM** 的版本，单位秒 s）：CM33 核从开始压缩到压缩完成的时间（用于去掉写SD卡的影响，来准确评估压缩耗时）。
7. **CM33** 核写 **SD** 卡耗时（单位秒 s）：计算方法 = 文件写入SD卡版本耗时 - 文件写入SRAM版本耗时。用于评估写 SD卡的耗时。
8. **HiFi4** 压缩耗时（FLAC写入**SRAM**版本，单位秒 s）：HiFi4 核从开始压缩到压缩完成的时间。最终压缩文件直接写入 SRAM。
9. **HiFi4** 压缩耗时比 **CM33** 压缩耗时：用于对比 HiFi4 与 CM33 压缩耗时的差异。
   计算公式：
   *Result = HiFi4 encode time consuming / CM33 encode time consuming * 100 %*
   上式耗时均指写入 SRAM 的版本。举例说明，如Table 6第一行，HiFi4 耗时比 CM33 = 0.060 / 0.242 * 100% ≈ 25%，表示 HiFi4 压缩耗时仅为 CM33 压缩耗时的 1/4，即减少了 ¾ 的时间。

Table 6. 压缩表现 **@16kHz_sine_mono_16bit.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	313	279	89 %	1.716	0.242	1.474	0.060	25 %
1	313	279	89 %	1.627	0.236	1.391	0.060	26 %
2	313	279	89 %	1.532	0.236	1.296	0.060	26 %
3	313	134	43 %	1.478	0.837	0.641	0.258	31 %

Table 6. 压缩表现 **@16kHz_sine_mono_16bit.wav***...continued*

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
4	313	123	39 %	1.534	0.997	0.537	0.311	31 %
5	313	123	39 %	1.549	0.999	0.550	0.311	31 %
6	313	118	38 %	2.275	1.780	0.495	0.553	31 %
7	313	115	37 %	2.865	2.421	0.444	0.759	31 %
8	313	100	32 %	4.039	3.642	0.397	1.121	31 %

Table 7. 压缩表现 **@48kHz_sine_mono_16bit.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	938	642	68 %	2.933	0.592	2.341	0.172	29 %
1	938	642	68 %	3.204	0.593	2.611	0.172	29 %
2	938	642	68 %	4.036	0.594	3.442	0.172	29 %
3	938	296	32 %	3.533	2.385	1.148	0.756	32 %
4	938	296	32 %	4.257	2.850	1.407	0.909	32 %
5	938	296	32 %	4.302	2.853	1.449	0.911	32 %
6	938	271	29 %	6.156	5.182	0.974	1.640	32 %
7	938	271	29 %	8.358	7.097	1.261	2.254	32 %
8	938	242	26 %	11.938	10.668	1.270	3.333	31 %

Table 8. 压缩表现 **@16kHz_speak_mono_16bit.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	313	228	73 %	1.464	0.230	1.234	0.059	26 %
1	313	228	73 %	1.491	0.232	1.259	0.059	26 %
2	313	228	73 %	1.400	0.233	1.167	0.059	25 %
3	313	228	73 %	1.882	0.848	1.034	0.260	31 %
4	313	227	73 %	1.666	1.013	0.653	0.315	31 %
5	313	226	72 %	1.822	1.029	0.793	0.315	31 %
6	313	223	71 %	2.575	1.827	0.748	0.560	31 %
7	313	222	71 %	3.273	2.486	0.787	0.775	31 %

Table 8. 压缩表现 **@16kHz_speak_mono_16bit.wav***...continued*

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
8	313	221	71 %	4.509	3.774	0.735	1.141	30 %

Table 9. 压缩表现 **@16kHz_speak_stereo_16bit.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	626	453	72 %	1.871	0.387	1.484	0.100	26 %
1	626	228	36 %	1.224	0.334	0.890	0.078	23 %
2	626	228	36 %	1.254	0.388	0.866	0.083	21 %
3	626	453	72 %	2.980	1.613	1.367	0.500	31 %
4	626	227	36 %	2.744	1.939	0.805	0.607	31 %
5	626	226	36 %	3.486	2.725	0.761	0.855	31 %
6	626	223	36 %	5.942	5.111	0.831	1.590	31 %
7	626	222	35 %	7.867	7.114	0.753	2.219	31 %
8	626	221	35 %	11.662	10.982	0.680	3.319	30 %

Table 10. 压缩表现 **@48kHz_speak_mono_16bit.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	938	569	61 %	2.806	0.576	2.230	0.165	29 %
1	938	569	61 %	2.811	0.577	2.234	0.165	29 %
2	938	569	61 %	2.784	0.577	2.207	0.165	29 %
3	938	551	59 %	4.209	2.420	1.789	0.764	32 %
4	938	549	59 %	4.686	2.908	1.778	0.923	32 %
5	938	548	58 %	5.079	2.912	2.167	0.925	32 %
6	938	547	58 %	7.780	5.282	2.498	1.654	31 %
7	938	545	58 %	9.439	7.294	2.145	2.292	31 %
8	938	544	58 %	13.563	11.113	2.450	3.383	30 %

Table 11. 压缩表现 **@48kHz_speak_stereo_16bit.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1876	1132	60 %	5.126	1.000	4.126	0.282	28 %
1	1876	570	30 %	3.029	0.843	2.186	0.220	26 %
2	1876	570	30 %	3.196	1.017	2.179	0.237	23 %
3	1876	1099	59 %	8.330	4.678	3.652	1.476	32 %
4	1876	550	29 %	6.121	4.174	1.947	1.327	32 %
5	1876	549	29 %	9.849	7.997	1.852	2.537	32 %
6	1876	547	29 %	17.049	15.110	1.939	4.725	31 %
7	1876	546	29 %	22.882	21.087	1.795	6.596	31 %
8	1876	545	29 %	34.503	32.521	1.982	9.873	30 %

Table 12. 压缩表现 **@48khz_mono_music_10s.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	940	699	74 %	4.419	0.598	3.821	0.168	28 %
1	940	699	74 %	3.242	0.595	2.647	0.168	28 %
2	940	699	74 %	3.185	0.595	2.590	0.168	28 %
3	940	667	71 %	4.489	2.448	2.041	0.770	31 %
4	940	658	70 %	5.135	2.956	2.179	0.932	32 %
5	940	658	70 %	5.776	2.948	2.828	0.934	32 %
6	940	658	70 %	7.586	5.330	2.256	1.665	31 %
7	940	648	69 %	9.393	7.357	2.036	2.312	31 %
8	940	648	69 %	14.396	11.233	3.163	3.410	30 %

Table 13. 压缩表现 **@48khz_stereo_music_10s.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time- write to SD card (s)	CM33 core compression time- write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time- write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1878	1411	75 %	7.925	1.029	6.896	0.289	28 %
1	1878	1387	74 %	6.877	1.100	5.777	0.286	26 %
2	1878	1381	74 %	5.993	1.272	4.721	0.308	24 %
3	1878	1351	72 %	9.105	4.724	4.381	1.489	32 %

Table 13. 压缩表现 **@48khz_stereo_music_10s.wav**...*continued*

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
4	1878	1308	70 %	11.754	6.743	5.011	2.123	31 %
5	1878	1307	70 %	14.731	10.588	4.143	3.344	32 %
6	1878	1306	70 %	24.373	20.131	4.242	6.270	31 %
7	1878	1288	69 %	32.455	28.198	4.257	8.801	31 %
8	1878	1288	69 %	47.913	43.703	4.210	13.192	30 %

Table 14. 压缩表现 **@48khz_stereo_music_sine_10s.wav**

FLAC compression level	Source file size (kB)	After encode file size (kB)	Compression rate	CM33 core compression time-write to SD card (s)	CM33 core compression time-write to SRAM (s)	Time consume of pure write data to SD card	HiFi4 core compression time-write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1878	1023	54 %	5.585	1.011	4.574	0.279	28 %
1	1878	1023	54 %	4.524	1.018	3.506	0.256	25 %
2	1878	1023	54 %	4.680	1.213	3.467	0.280	23 %
3	1878	912	49 %	8.972	4.660	4.312	1.472	32 %
4	1878	905	48 %	10.673	6.636	4.037	2.078	31 %
5	1878	905	48 %	13.314	10.471	2.843	3.301	32 %
6	1878	887	47 %	23.087	19.970	3.117	6.224	31 %
7	1878	877	47 %	31.792	27.891	3.901	8.724	31 %
8	1878	858	46 %	47.353	43.080	4.273	13.099	30 %

通过 Table 6 到 Table 14，可以得到下面几个结论：

关于 FLAC 的压缩表现：

1. 通过 Table 6 到 Table 14，我们可以明显的看出：如果逐步提高压缩等级，压缩率也会逐步提升（数值降低），但压缩耗时会逐步增大，这也与实际感受情况一致。
2. 通过 Table 6 到 Table 14，对比“CM33核压缩耗时（写入 SD 卡版本）” 和“CM33 核压缩耗时(写入 SRAM 版本)”两项，我们可以看出耗时差异很大。这验证了我们的猜想：如果需要准确的评估 FLAC 压缩的实际耗时，需要去除读写 SD 的影响。同时我们也发现CM33写SD卡的耗时很不稳定，即使同样大小的文件（如Table 7的level 0-2），也有几百毫秒的差异。初步判断是 SD 卡文件系统存储机制（如 ECC,读写均衡）导致。
3. 通过 Table 6 和 Table 7，将待压缩音频的采样率由 16 KHz提高到 48 KHz，对比 CM33 核和 HiFi4 核的压缩耗时，也均是几乎增加了 3 倍。所以证明采样率的提升与 FLAC 压缩耗时是线性正相关。
4. 通过 Table 6 和 Table 8，以及Table 7，Table 10，和Table 12可以得出，对于仅音频内容不同（如 sine，speak和Music，均为mono）的数据，FLAC 同等级encode耗时几乎一致，压缩耗时差异5%以内。但是压缩率是有差异的，明显 Sine wave 的压缩率会

优于 Speak 对话的音频。这也与实际情况一致。因为正弦波信号更容易预测得到更小的残差，最终 Rice 编码值也会更小。

5. 通过 Table 8 和 Table 9，以及 Table 10 和 Table 11可以得出，对于同一个音频仅 mono 和 stereo 的区别，FLAC 压缩耗时并不是 2 倍的关系。越高的压缩率 stereo 会引入更复杂的线性预测编码（LPC）运算，双通道音频比单通道更耗时，且大于两倍。
6. 通过 Table 11 和 Table 13 可以得出，对于stereo音频，两通道数据不相同会比两通道相同的音频更耗时，耗时增加 30 %左右。
7. 通过 Table 13 和 Table 14 可以得出，对于两通道数据都不相同的stereo音频，仅音频内容有差异，最终压缩耗时相差不大。与本节结论4吻合。
8. 通过 Table 9 和 Table 11 可以看到对于stereo音频，第 3 压缩等级有一个异常的表现。其压缩率还不如低压缩等级的 1 和 2。通过对比 Table 4 可以发现是因为压缩率 3 没有使用*do_mid_side_stereo*选项导致。所以可以得出对于 Stereo 音频，我们要尽量选择开启该选项的的压缩等级。

关于 RT685 的压缩表现：

1. 通过 Table 6 到 Table 14，我们可以明显的看出：在 HiFi4 运行频率 600 MHz 对比 CM33 运行频率 300 MHz 的情况下，HiFi4 运算性能明显优于 CM33，且速度提升大于两倍。HiFi4 核压缩耗时平均仅为 CM33 核压缩耗时的 1/3。
2. 通过 Table 6 到 Table 14，我们可以看出对于主频在 300 MHz 的 CM33 核来说：
   a. 对于低音质的音频，如16KHz_mono_16bit这类（Table 6 和 Table 8）：0-8 所有的压缩等级都能做到实时压缩（我们这里将1 s原始音频可以在 1 s 内完成压缩，即认为可以实时压缩）。
   b. 对于 48KHz_mono_16bit，16KHz_stereo_16bit 这类（Table 7，Table 9，Table 10和 Table 12）：除了最高等级 8 不能做到实时压缩，其他低于 8 压缩等级的都是可以满足的。
   c. 对于 48KHz_stereo_16bit 这类高保真音频（Table 11，Table 13，和 Table 14）：5 及以下的压缩率能够实时压缩.
3. 通过 Table 6 到 Table 14，我们可以看出对于主频在 600 MHz 的 HiFi4 DSP 核来说：
   - 仅对 48KHz_stereo_16bit 这类双通道不完全相同的音频在 8 压缩等级稍显吃力，对其他类型的音频都额能够做到0-8 级压缩率的实时压缩。且耗时仅为 CM33 的25 % - 30 %，处理音频数据效率更高。

6.2 解压表现

对于 Section 5.2 中的9个文件，经过 Section 6.1 的压缩，可以得到 9*9=81 个 FLAC 文件。我们将其依次在 CM3 3和 HiFi4 两个核上进行解压。得到如下 9 个表。每个表除去 Title 行外由 9 行构成，每行代表一个压缩等级。每一列含义依次是：

1. 压缩等级：范围 0-8，FLAC 预先定义的压缩等级。
2. **FLAC** 原始文件大小（单位 kB）：这里是使用对应压缩等级，压缩后生成FLAC的文件大小。
3. 解压后文件大小（单位 kB）：这里是解压后WAV文件的大小。
4. 与原始 **WAV** 文件是否一致：将解压后的 WAV文件 与原始 Section 5.2 的 6 个 WAV 文件对比，如果完全一致就是 Yes，否则为 No。
5. **CM33** 核解压耗时（WAV 文件写入 **SD** 卡的版本，单位秒 s）：CM33 核从开始解压到解压完成的时间（用于获取压缩后的文件验证压缩正确性）。
6. **CM33** 核解压耗时（WAV 文件写入 **SRAM** 的版本，单位秒 s）: CM33 核从开始解压到解压完成的时间（用于去掉写SD卡的影响，来准确评估解压耗时）。
7. **CM33** 核写 **SD** 卡耗时（单位秒 s）：计算方法 = 文件写入 SD 卡版本耗时 - 文件写入 SRAM 版本耗时。

8. **HiFi4** 解压写入 **SRAM** 耗时（WAV 文件写入 **SRAM** 的版本，单位秒 s）：HiFi4 核从开始解压到解压完成的时间（用于去掉写 SD 卡的影响，来准确评估解压耗时）。
9. **HiFi4** 解压耗时比 **CM33** 解压耗时：用来评估HiFi4解压耗时是CM33解压耗时的多少倍。
   计算公式：
   解压耗时比 = HiFi4 解压耗时/CM33 解压耗时 * 100 %
   以上解压耗时都使用写入 SRAM 的版本来计算。举例说明，如下表第一行，耗时对比 = 0.025/0.099 * 100 % ≈ 25 %，这表示完成同样的解压任务，HiFi4 只需要相当于 CM33 ¼ 的时间。

Table 15. 解压表现 **@16kHz_mono_sine_16bit.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	279	313	Yes	2.144	0.099	2.045	0.025	25 %
1	279	313	Yes	2.084	0.092	1.992	0.025	27 %
2	279	313	Yes	1.876	0.095	1.781	0.025	26 %
3	134	313	Yes	1.762	0.094	1.668	0.025	26 %
4	123	313	Yes	1.765	0.101	1.665	0.027	27 %
5	123	313	Yes	1.776	0.096	1.680	0.027	28 %
6	118	313	Yes	1.767	0.095	1.672	0.025	26 %
7	115	313	Yes	1.767	0.096	1.671	0.027	28 %
8	100	313	Yes	1.866	0.098	1.768	0.026	26 %

Table 16. 解压表现 **@48kHz_mono_sine_16bit.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	642	938	Yes	6.635	0.247	6.388	0.066	27 %
1	642	938	Yes	7.126	0.250	6.876	0.066	27 %
2	642	938	Yes	5.764	0.247	5.517	0.066	27 %
3	296	938	Yes	5.485	0.241	5.244	0.059	25 %
4	296	938	Yes	5.588	0.238	5.350	0.059	25 %
5	296	938	Yes	5.574	0.241	5.333	0.059	25 %
6	271	938	Yes	5.677	0.261	5.416	0.059	23 %
7	271	938	Yes	5.599	0.241	5.358	0.059	25 %
8	242	938	Yes	5.397	0.239	5.158	0.059	25 %

Table 17. 解压表现 **@16kHz_mono_speak_16bit.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	228	313	Yes	1.782	0.094	1.688	0.025	26 %
1	228	313	Yes	1.776	0.091	1.685	0.025	27 %
2	228	313	Yes	1.797	0.094	1.703	0.025	26 %
3	228	313	Yes	1.961	0.096	1.865	0.026	27 %
4	227	313	Yes	1.991	0.104	1.887	0.028	27 %
5	226	313	Yes	2.075	0.102	1.973	0.028	27 %
6	223	313	Yes	2.160	0.106	2.054	0.029	27 %
7	222	313	Yes	2.126	0.113	2.013	0.037	32 %
8	221	313	Yes	1.941	0.116	1.825	0.036	31 %

Table 18. 解压表现 **@16kHz_stereo_speak_16bit.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	453	626	Yes	3.790	0.163	3.627	0.043	26 %
1	228	626	Yes	4.432	0.137	4.295	0.039	28 %
2	228	626	Yes	4.298	0.139	4.159	0.037	26 %
3	453	626	Yes	3.738	0.176	3.562	0.045	25 %
4	227	626	Yes	3.691	0.144	3.547	0.043	30 %
5	226	626	Yes	4.306	0.145	4.161	0.041	28 %
6	223	626	Yes	3.570	0.143	3.427	0.041	28 %
7	222	626	Yes	3.487	0.156	3.331	0.049	31 %
8	221	626	Yes	3.966	0.153	3.813	0.048	31 %

Table 19. 解压表现 **@48kHz_mono_speech_10s.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original .wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	569	938	Yes	5.613	0.245	5.369	0.061	25 %
1	569	938	Yes	5.766	0.242	5.524	0.061	25 %

Table 19. 解压表现 **@48kHz_mono_speech_10s.wav**...*continued*

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original .wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
2	569	938	Yes	5.642	0.245	5.397	0.061	25 %
3	551	938	Yes	5.873	0.257	5.616	0.064	25 %
4	549	938	Yes	5.944	0.273	5.671	0.070	26 %
5	548	938	Yes	5.571	0.271	5.300	0.070	26 %
6	547	938	Yes	5.280	0.276	5.004	0.071	26 %
7	545	938	Yes	5.772	0.322	5.450	0.091	28 %
8	544	938	Yes	6.003	0.300	5.703	0.090	30 %

Table 20. 解压表现 **@48kHz_stereo_speak_16bit.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1132	1876	Yes	11.240	0.451	10.789	0.115	25 %
1	570	1876	Yes	11.506	0.371	11.136	0.104	28 %
2	570	1876	Yes	11.406	0.360	11.046	0.098	27 %
3	1099	1876	Yes	12.766	0.493	12.273	0.124	25 %
4	550	1876	Yes	11.800	0.396	11.404	0.114	29 %
5	549	1876	Yes	11.256	0.392	10.864	0.108	28 %
6	547	1876	Yes	11.692	0.391	11.301	0.108	28 %
7	546	1876	Yes	10.812	0.420	10.392	0.128	30 %
8	545	1876	Yes	11.526	0.419	11.107	0.128	30 %

Table 21. 解压表现 **@48khz_mono_music_10s.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original .wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	699	940	Yes	5.860	0.253	5.607	0.061	24 %
1	699	940	Yes	6.318	0.251	6.067	0.061	25 %
2	699	940	Yes	5.564	0.275	5.289	0.061	22 %
3	667	940	Yes	5.271	0.275	4.997	0.068	25 %

Table 21. 解压表现 **@48khz_mono_music_10s.wav**...*continued*

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original .wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
4	658	940	Yes	5.298	0.292	5.006	0.076	26 %
5	658	940	Yes	5.325	0.290	5.035	0.076	26 %
6	658	940	Yes	5.542	0.293	5.250	0.076	26 %
7	648	940	Yes	5.318	0.319	4.999	0.104	33 %
8	648	940	Yes	5.318	0.322	4.997	0.104	32 %

Table 22. 解压表现 **@48khz_stereo_music_10s.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1411	1878	Yes	10.611	0.463	10.148	0.116	25 %
1	1387	1878	Yes	10.387	0.498	9.889	0.136	27 %
2	1381	1878	Yes	10.451	0.495	9.956	0.135	27 %
3	1351	1878	Yes	10.420	0.517	9.903	0.131	25 %
4	1308	1878	Yes	10.482	0.580	9.902	0.167	29 %
5	1307	1878	Yes	10.489	0.583	9.906	0.167	29 %
6	1306	1878	Yes	10.468	0.582	9.886	0.167	29 %
7	1288	1878	Yes	11.308	0.644	10.664	0.225	35 %
8	1288	1878	Yes	10.934	0.642	10.293	0.223	35 %

Table 23. 解压表现 **@48khz_stereo_music_sine_10s.wav**

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
0	1023	1878	Yes	10.552	0.452	10.100	0.114	25 %
1	1023	1878	Yes	10.357	0.475	9.882	0.114	24 %
2	1023	1878	Yes	10.365	0.453	9.913	0.114	25 %
3	912	1878	Yes	10.367	0.481	9.887	0.121	25 %
4	905	1878	Yes	10.441	0.499	9.942	0.130	26 %
5	905	1878	Yes	10.390	0.500	9.890	0.130	26 %

Table 23. 解压表现 @48khz_stereo_music_sine_10s.wav...*continued*

Compression level	FLAC Source file size (kB)	After decode file size (kB)	consistent with the original. wav file	CM33 core decode time - write to SD (s)	CM33 core decode time - write to SRAM (s)	Time consume of pure write data to SD card (s)	HiFi4 core decode time - write to SRAM (s)	HiFi4 time consuming/ CM33 time consuming
6	887	1878	Yes	10.489	0.495	9.994	0.128	26 %
7	877	1878	Yes	10.416	0.529	9.887	0.157	30 %
8	858	1878	Yes	10.472	0.531	9.941	0.157	30 %

通过 Table 15 到 Table 23，可以得到下面几个结论：

关于 **FLAC** 的解压缩表现：

1. 我们可以看到对于同一文件在不同压缩等级得到的压缩音频，其解压耗时基本一致。
2. 对比编码，FLAC 压缩等级只对编码时间有较大的影响，而对解码影响不大。可以理解为：压缩等级越高，编码器就需要花越多的时间去寻找最佳的压缩算法（阶数更高），而解码器则只需要根据已知的压缩算法参数直接解压即可。
3. FLAC 算法设计的思路就是方便解压。目的是便与解码器在不同档次的硬件上实现，这有助于 FLAC 编码的推广。

关于 **RT685** 的解压缩表现：

1. 通过 Table 15 到 Table 23，我们可以明显的看出：在 HiFi4 运行频率 600 MHz 对比 CM33 运行频率 300 MHz 的情况下，HiFi4 运算性能明显优于 CM33，且速度提升大于两倍。HiFi4 核解压耗时平均仅为 CM33 核解压耗时的1/3 ~ 1/4。这与 Section 6.1 中的压缩结论基本一致。
2. 通过 Table 15 到 Table 23，我们可以看出对于主频 300 MHz 的 CM33 和 600 MHz 的 HiFi4 DSP 来说，都能够在 0.6 s 内将 Table 5 中压缩后的 10 s FLAC 音频解压为原始的 WAV 数据。

6.3 FLAC 内存资源消耗

以下评估 FLAC 编解码对 MCU 资源的消耗基于 IAR（配置见 Table 5）。方法如下：对于 TEXT 和 DATA，我们可以对比加入解码前后的 map 文件，找到差值即可。对于 HEAP 区域由于是动态变化的，我们可以使用 `__iar_dlmallinfo()` 函数采样获取 heap 的统计信息。对于 Stack 区域，可以通过使能 IAR 的 “Enable stack usage analysis” 获取，结果如 Table 24 所示。特别需要注意的是 FLAC 压缩大于 2 级后，由于会使用线性预测算法，Heap 消耗会增加很多。

Table 24. 资源小号表

Category	ENCODER		DECODER
Compress level	0-2	3-8	0-8
RO TEXT	95’578 bytes		35’638 bytes
RO DATA	14’832 bytes		8’328 bytes
R/W DATA	556 bytes		552 bytes
Heap	158’576 bytes	431’360 bytes	99’784 bytes
Stack	6’232 bytes		5’272 bytes

7 GNU profiler 分析

7.1 GNU profiler 介绍

Profiler 是 GNU 的一个工具，用来分析用户代码中每个函数的耗时占比，函数调用信息等。它可以很方便的帮我们找出最耗时的函数，或者预料外的函数调用关系，这可以帮助我们快速调试和定位问题。

Tensilica Xtensa 支持硬件 profiling 方式。其原理是将待评估程序在对应的硬件平台上运行，然后再通过一个 Xtensa 定时器定时采样，然后统计各个函数的耗时。最后调试器将生成的 profiling 数据通过 OCD Deamon 工具发送给 PC 端存储起来（依赖 libgdbio 库，所以必须使用 gdbio LSP 来链接执行文件）。想要进行 Tensilica 的硬件 profiling 分析，需要以下三个步骤：

- 源码编译和链接添加配置使能 `profiling`。
- 添加 profile 代码并运行。
- 使用 gprof 解析原始 `gmon.out`。

更多详细步骤，可参考 Xtensa 软件安装目录下的《GNU Profiler User's Guide》文档。

7.2 Profiling 文件分析

通过 Section 7.1 中的 3 个步骤，我们可以得到如 Figure 7 的 profile.txt 文件。其包括 Flat profile，Call Graph，annotated source 几个部分。在 Flat profile 单元，每个函数是按照耗时进行递减排序的，最前面的就是最耗时的，反之亦然。所以我们可以看到对于 FLAC encoder 程序来说，比较耗时的 FLAC__lpc_compute_autocorrelation 函数，那么对于下一步的优化我们也就有了明确的方向。比如将耗时的函数放到 HiFi4 的 ITCM 区域运行，或者使用 HiFi 4 特殊指令集重写这几个函数都是不错的方法。

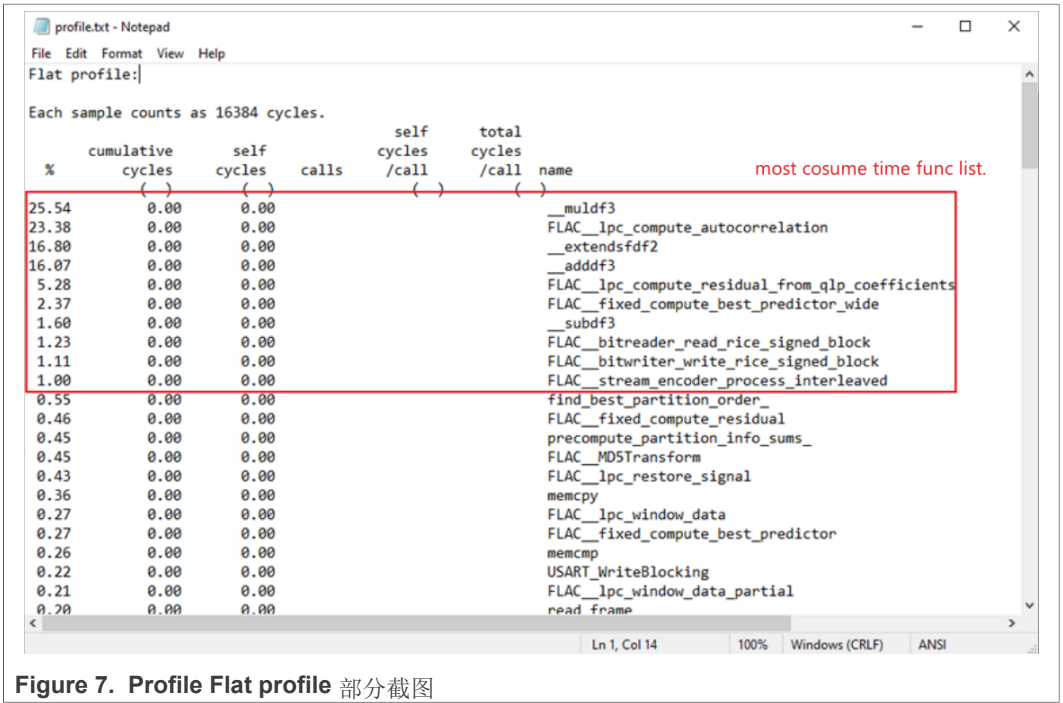

Figure 7. Profile Flat profile 部分截图

8 结论

1. 验证了开源库 FLAC 是无损的编解码。
2. 对于 FLAC 解压，RT685 的 CM33 和 HiFi4 DSP 两个核都能轻松胜任。
3. 对于 FLAC 压缩，RT685 的 CM33 和 HiFi4 对于大部分音频都能够实现 0-8 全等级的实时压缩，对少部分高质量音频可实现稍低等级的实时压缩。
4. 在编解码方面，HiFi 4 DSP@600 MHz 耗时均仅为 CM33@300 MHz 的 25 - 30 %。
5. 对比解压，压缩的不同压缩等级对运行耗时影响很大。项目使用时需要合理选择最优的压缩等级，平衡时间与空间。

9 参考资料

1. FLAC website
2. GNU gprof
3. GNU Profiler User's Guide Version 2.34
4. MIMXRT600-EVK Schematic (Rev E2)
5. *RT600 User Manual* (document UM11147)

10 修订历史

版本号	日期	说明
0	2022 年 11 月 30 日	初次发布

11 Legal information

11.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

11.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

11.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: http://www.nxp.com

Date of release: 30 November 2022
Document identifier: AN13793