



客户端 AMR 编解码库

API 参考

文档版本 03

发布日期 2008-11-30

BOM编码 N/A

秘密

版权所有 © 深圳市海思半导体有限公司

深圳市海思半导体有限公司为客户提供全方位的技术支持，用户可与就近的海思办事处联系，也可直接与公司总部联系。

深圳市海思半导体有限公司

地址： 深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址： <http://www.hisilicon.com>

客户服务电话： 0755-28788858

客户服务传真： 0755-28357515

客户服务邮箱： support@hisilicon.com.

版权所有 © 深圳市海思半导体有限公司 2007-2008。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思，均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

秘密

版权所有 © 深圳市海思半导体有限公司



目 录

前 言.....	1
1 概述.....	1-1
1.1 AMR-NB	1-1
1.2 数据类型	1-2
1.3 函数列表	1-2
1.4 参考信息分类	1-3
1.5 参考关联信息	1-3
1.6 函数描述方式	1-3
1.7 结构体描述方式	1-4
2 API参考	2-1
2.1 初始化函数	2-1
2.2 应用执行函数	2-5
3 其他信息.....	3-1
3.1 数据类型	3-1
3.2 错误码	3-2
A 缩略语	A-1



表格目录

表 1-1 数据类型表.....	1-2
表 1-2 编码软件包定义函数.....	1-2
表 1-3 解码软件包定义函数.....	1-3



前 言

概述

本节介绍本文档的内容、对应的产品版本、适用的读者对象、行文表达约定、历史修订记录等。

产品版本

与本文档相对应的产品版本如下所示。

产品名称	产品版本
Hi3510 通信媒体处理器	V100
Hi3511 H.264 编解码处理器	V100
Hi3512 H.264 编解码处理器	V100

读者对象

本文档适合下列人员阅读：

- 软件开发人员
- 技术支持人员

内容简介

本文档介绍了 AMR 编解码库的相关内容。首先概述 AMR 编解码库的 API 参考种类及其间关联，然后分别详细介绍了各种参考信息。

章节	内容
1 概述	介绍 AMR 命令接口 API 的参考信息分类、各类参考信息的作用和关联。函数及数据结构的描述方式等。
2 API 参考	首先根据功能分类列出全部 API，然后详细介绍每一个 API 接口函数。





章节	内容
3 其他信息	介绍除上述参考项外的其他信息参考，包括数据结构、错误码等。
A 缩略语	介绍本文档中出现的缩略语。

约定

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	以本标志开始的文本表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	以本标志开始的文本表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	以本标志开始的文本表示有潜在风险，如果忽视这些文本，可能导致设备或器件损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	以本标志开始的文本能帮助您解决某个问题或节省您的时间。
 说明	以本标志开始的文本是正文的附加信息，是对正文的强调和补充。

通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。
楷体	警告、提示等内容一律用楷体，并且在内容前后增加线条与正文隔离。
“Terminal Display” 格式	“Terminal Display” 格式表示屏幕输出信息。此外，屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。



修改记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修改日期	版本	修改说明
2008-09-05	03	增加本文档适用的 Hi3512 产品版本。
2008-04-21	02	增加本文档适用的 Hi3511 产品版本。
2007-04-05	01	第 1 次发布



1 概述

1.1 AMR-NB

AMR-NB (Adaptive Multi-Rate Narrow-Band) 语音编解码标准是由 3GPP (3rd Generation Partnership Project) 提出的, 主要应用于 3G 移动语音设备。

AMR-NB 主要功能特性:

- 支持 8 种基本速率, 分别为 12.2/10.2/7.95/7.40/6.70/5.90/5.15/4.75 kbit/s。所有速率均基于 ACELP (Algebraic Code Excited Linear Prediction) 编码。
- 为降低码率, AMR-NB 支持非连续传输 DTX (Discontinuous Transmission)、语音检测 VAD (Voice Activity Detection) 和舒适噪声生成 CNG (Comfort Noise Generation), 采用 VAD1 语音检测算法。
- 支持 8k 采样, 20ms (160 个采样点) 帧长处理。
- 支持 3 种文件存储格式:
 - MIME
RTP Payload format, 请参见 “rfc4239 AMR and AMR-WB Storage Format”。
 - IF1
请参见 “3GPP 26101-600.doc”。
 - IF2
请参见 “3GPP 26101-600.doc”。

AMR-NB 在标准方面:

- 符合以下 3GPP GSM-AMR 标准。
 - 3GPP TS 26.071 V4.0.0 AMR Speech Codec; General Description
 - 3GPP TS 26.090 V4.0.0 AMR Speech Codec; Transcoding functions
 - 3GPP TS 26.091 V4.0.0 AMR Speech Codec; Error concealment of lost frames
 - 3GPP TS 26.092 V4.0.0 AMR Speech Codec; Comfort noise aspects
 - 3GPP TS 26.093 V4.0.0 AMR Speech Codec; Source controlled rate operation
 - 3GPP TS 26.094 V4.0.0 AMR Speech Codec; Voice activity detector
- 与 3GPP GSM-AMR 标准参考代码比特一致。
3GPP TS 26.071 V4.0.0 ANSI-C code for AMR speech codec(Code Version 7.5.0)。



- 针对 3GPP GSM-AMR 标准所提供的测试码流进行了完整测试。
3GPP TS 26.074 V4.0.0 AMR Speech Codec, Test sequences。
- 速率的选择及 DTX 的开关可以帧为单位进行。

1.2 数据类型

软件包使用的全部数据类型如表 1-1 所示。

表1-1 数据类型表

数据类型	描述
HI_U8	8-bit 无符号字符类型。
HI_S8	8-bit 有符号字符类型。
HI_U16	16-bit 无符号整数类型。
HI_S16	16-bit 有符号整数类型。
HI_U32	32-bit 无符号整数类型。
HI_S32	32-bit 有符号整数类型。
HI_VOID	void 类型。

1.3 函数列表

编码软件包所定义的函数如表 1-2 所示。

表1-2 编码软件包定义函数

函数	功能
AMR_Encode_Init	初始化编码设备。
AMR_Encode_Frame	编码。
AMR_Encode_Exit	释放编码设备。

解码软件包所定义的函数如表 1-3 所示。



表1-3 解码软件包定义函数

函数	功能
AMR_Decode_Init	初始化解码设备。
AMR_Get_Length	计算帧长度，以 HI_U8 为单位。
AMR_Decode_Frame	解码。
AMR_Decode_Exit	释放解码设备。

1.4 参考信息分类

驱动软件 API 参考信息分为以下 2 类。

种类	作用
API 参考	详细介绍每个 API 接口函数。
其他信息	介绍除上述参考项外的其他信息参考，如数据结构、错误码等。

1.5 参考关联信息

API 中涉及的数据结构和错误码定义在“[3 其他信息](#)”相应的部分统一描述。

1.6 函数描述方式

本文档对 API 参考信息使用以下域来描述。

参数域	作用
目的	简要描述 API 的主要功能。
语法	显示 API 的语法样式。
描述	简要描述本 API 的工作过程。
参数	列出 API 的参数、参数说明及其属性。
返回值	列出 API 的返回值及其返回值说明。
错误码	列出 API 的错误码及其错误码说明。
需求	列出本 API 要包含的头文件和 API 编译时要链接的库文件。



参数域	作用
注意	使用 API 时应注意的事项。
举例	使用 API 的实例。
相关主题	同本 API 的其他相关信息。

1.7 结构体描述方式

本参考对结构体使用以下域来描述，它们的作用如下所示。

参数域	作用
说明	简单描述结构体所实现的功能。
定义	列出结构体的定义。
注意	列出结构体的注意事项。



2 API 参考

2.1 初始化函数

AMR_Encode_Init

【目的】

开始编码任务时，初始化编码设备。

【语法】

```
#include "amr_enc.h"
HI_S32 AMR_Encode_Init (HI_VOID **pEncState, HI_S16 dtx);
```

【描述】

根据上层传入的编码器设备指针，初始化编码设备。DTX 控制开关信息由用户指定。若 DTX 打开，声码器进行语音检测活动，若检测到静音帧，则降低传输速率，以降低用户功耗，提高网络容量。

【参数】

参数名称	描述	输入/输出	全局/局部
pEncState	用户指定的编码设备。	输入/输出	全局
dtx	DTX 使能标志。 0：不使能。 1：使能。	输入	局部

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。



【错误码】

错误码	描述
HI_ERR_AMRNB_INIT_FAIL	编码设备初始化失败。

【需求】

- 头文件: /include/amr_enc.h
- 库文件: /lib/amrnb.lib

【注意】

无

【举例】

```
#include "amr_enc.h"
HI_VOID *pEncState = NULL;
HI_S16 dtx = 0;
if (AMR_Encode_Init(&pEncState, dtx))
{
    fprintf(stderr, "\nerror AMR_Encode_Init fail: %s\n",strerror(errno));
    exit(-1);
}
```

【相关主题】

- HI_S32 AMR_Encode_Frame
- HI_VOID AMR_Encode_Exit

AMR_Encode_Exit

【目的】

编码任务结束后，释放编码设备。

【语法】

```
#include "amr_enc.h"
HI_VOID AMR_Encode_Exit (HI_VOID **pEncState);
```

【描述】

无

【参数】

参数名称	描述	输入/输出	全局/局部
pEncState	用户指定的编码设备。	输入	全局



【返回值】

无

【需求】

- 头文件: /include/amr_enc.h
- 库文件: /lib/amrnb.lib

【注意】

无

【举例】

```
#include "amr_enc.h"
HI_VOID *pEncState = NULL;
AMR_Encode_Exit(&pEncState);
```

【相关主题】

- HI_S32 AMR_Encode_Init
- HI_S32 AMR_Encode_Frame

AMR_Decode_Init

【目的】

开始解码任务时，初始化解码设备。

【语法】

```
#include "amr_dec.h"
HI_S32 AMR_Decode_Init (HI_VOID **pDecState);
```

【描述】

无

【参数】

参数名称	描述	输入/输出	全局/局部
pDecState	用户指定的解码设备。	输入/输出	全局

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。



【错误码】

错误码	描述
HI_ERR_AMRNB_INIT_FAIL	解码设备初始化失败。

【需求】

- 头文件: /include/amr_dec.h
- 库文件: /lib/amrnb.lib

【注意】

无

【举例】

```
#include "amr_enc.h"
HI_VOID *pDecState = NULL;
if (AMR_Decode_Init(&pDecState))
{
    fprintf(stderr, "\nerror AMR_Decode_Init fail: %s\n",strerror(errno));
    exit(-1);
}
```

【相关主题】

- HI_S32 AMR_Decode_Frame
- HI_VOID AMR_Decode_Exit

AMR_Decode_Exit

【目的】

解码任务结束后，释放解码设备。

【语法】

```
#include "amr_dec.h"
HI_VOID AMR_Decode_Exit (HI_VOID ** pDecState);
```

【描述】

无

【参数】

参数名称	描述	输入/输出	全局/局部
pDecState	用户指定的解码设备。	输入	全局



【返回值】

无

【需求】

- 头文件: /include/amr_dec.h
- 库文件: /lib/amrnb.lib

【注意】

无

【举例】

```
#include "amr_dec.h"
HI_VOID *pDecState = NULL;
AMR_Decode_Exit(&pDecState);
```

【相关主题】

- HI_S32 AMR_Decode_Init
- HI_S32 AMR_Decode_Frame

2.2 应用执行函数

AMR_Encode_Frame

【目的】

对原始语音数据进行编码。

【语法】

```
#include "amr_enc.h"
HI_S32 AMR_Encode_Frame (HI_VOID *pEncState,
HI_S16 *pInBuf,
HI_U8 *pOutBuf,
enum Mode mode,
enum Format frame_type);
```

【描述】

用户传入待编码语音数据，保证输入数据长度为 L_FRAME（160，以 HI_S16 为单位）。编码器根据 enum Mode 变量所指定速率进行编码，编码后根据 enum Format 变量所指定帧类型将编码后的数据打包存放在用户指定的输出缓冲区，并返回编码后数据长度（以 HI_U8 为单位）。

【参数】



参数名称	描述	输入/输出	全局/局部
pEncState	用户指定的编码设备。	输入/输出	全局
pInBuf	待编码数据输入缓冲。	输入	局部
pOutBuf	已编码数据输出缓冲。	输出	局部
mode	编码基本速率。	输入	局部
frame_type	帧格式。	输入	局部

【返回值】

返回值	描述
正值	成功，其值为编码后数据长度，以 HI_U8 为单位。
负值	失败，其值为错误码。

【错误码】

错误码	描述
HI_ERR_AMRNB_INVALID_DEVICE	非法输入编码设备指针。
HI_ERR_AMRNB_INVALID_INBUF	非法输入缓冲指针。
HI_ERR_AMRNB_INVALID_OUTBUF	非法输出缓冲指针。
HI_ERR_AMRNB_MODE_TYPE	非法输入编码速率。
HI_ERR_AMRNB_FORMAT_TYPE	非法输入帧格式。
HI_ERR_AMRNB_ENCODE_FAIL	编码失败。

【需求】

- 头文件：/include/amr_enc.h
- 库文件：/lib/amrnb.lib

【注意】

- 保证编码输入数据长度为 L_FRAME（160，以 HI_S16 为单位）。
- 保证输出缓冲区不小于 MAX_PACKED_SIZE（35，以 HI_U8 为单位）。

【举例】

```
#include "amr_enc.h"
HI_S16 pInBuf[L_FRAME];          /*L_FRAME 输入数据长度,160*/
```



```
/*MAX_PACKED_SIZE 为编码后最大数据长度（以HI_U8为单位）*/
HI_U8 pOutBuf[MAX_PACKED_SIZE];
HI_VOID *pEncState = NULL;
HI_S32 packed_size;
HI_S16 dtx = 1;
enum Mode mode = MR122;
enum Format frame_type = MIME;

if (AMR_Encode_Init(&pEncState, dtx))
{
    fprintf(stderr, "\nerror AMR_Encode_Init fail: %s\n", strerror(errno));
    exit(-1);
}

If (frame_type == MIME) /*本地存储时，若帧格式为MIME，则需要写入文件头*/
{
    fwrite(AMR_MAGIC_NUMBER, sizeof(HI_S8), strlen(AMR_MAGIC_NUMBER),
        file_out);
}

while(fread(pInBuf, sizeof(HI_S16), L_FRAME, file_in) == L_FRAME)
{
    packed_size=
        AMR_Encode_Frame (pEncState, pInBuf, pOutBuf, mode, frame_type);
    if (packed_size < 0)
    {
        fprintf(stderr, "\nerror AMR_Encode_Frame fail:
            %s\n", strerror(errno));
        exit(-1);
    }
    fwrite (pOutBuf, sizeof(HI_U8), packed_size, file_out);
}
AMR_Encode_Exit(&pEncState);
```

【相关主题】

- HI_S32 AMR_Encode_Init
- HI_VOID AMR_Encode_Exit

AMR_Get_Length

【目的】

由帧头信息，计算该帧数据长度。以 HI_U8 为单位。

【语法】

```
HI_S32 AMR_Get_Length(enum Format frame_type, HI_U8 toc)
```

**【描述】**

AMR-NB 支持 8 种速率编解码，3 种帧格式，因此编码后数据长度不固定。解码前需根据帧头信息计算出该帧数据长度以正确取码流。

【参数】

参数名称	描述	输入/输出	全局/局部
frame_type	帧格式。	输入	局部
toc	帧头。	输入	局部

【返回值】

返回值	描述
非负	成功，其值为该帧码流长度减 1。以 HI_U8 为单位。
负值	失败，其值为错误码。

【错误码】

错误码	描述
HI_ERR_AMRNB_FORMAT_TYPE	非法帧格式。

【需求】

- 头文件：/include/amr_dec.h
- 库文件：/lib/amrnb.lib

【注意】

- 保证编码和解码帧格式相同，即 enum Format 变量保持一致。
- AMR-NB 解码需先读取帧头（1byte），再根据帧头信息计算帧长度。由于此前已经读取 1byte，因此 AMR_Get_Length 返回值为该帧码流长度减 1。

【举例】

```
#include "amr_dec.h"
HI_U8 pInBuf[MAX_PACKED_SIZE];
HI_S32 packed_size;

fread(&pInBuf[0], sizeof(HI_U8), 1, file_in); /*获得帧头信息，放入输入缓冲区*/
packed_size = AMR_Get_Length(MIME, pInBuf[0]);
```

【相关主题】

HI_S32 AMR_Decode_Frame



AMR_Decode_Frame

【目的】

解码器对输入的一帧数据解码，输出解码语音数据。

【语法】

```
#include "amr_dec.h"

HI_S32 AMR_Decode_Frame(HI_VOID *pDecState,
HI_U8 *pInBuf,
HI_S16 *pOutBuf,
enum Format frame_type);
```

【描述】

用户传入编码压缩码流，解码器解码后，把已解码的声音数据存放在用户指定的输出缓冲区。解码后数据长度为 160（HI_S16）。

【参数】

参数名称	描述	输入/输出	全局/局部
pDecState	用户指定的解码设备。	输入/输出	全局
pInBuf	待解码数据输入缓冲。	输入	局部
pOutBuf	解码数据输出缓冲。	输出	局部
frame_type	帧格式。	输入	局部

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

错误码	描述
HI_ERR_AMRNB_INVALID_DEVICE	非法输入编解码设备指针。
HI_ERR_AMRNB_INVALID_INBUF	非法输入缓冲指针。
HI_ERR_AMRNB_INVALID_OUTBUF	非法输出缓冲指针。
HI_ERR_AMRNB_FORMAT_TYPE	非法帧格式。
HI_ERR_AMRNB_DECODE_FAIL	解码失败。

**【需求】**

- 头文件: /include/amr_dec.h
- 库文件: /lib/amrnb.lib

【注意】

- 保证编码和解码帧格式相同, 即 enum Format 变量保持一致。
- 保证解码输入缓冲区不小于 MAX_PACKED_SIZE (35, 以 HI_U8 为单位)。
- 保证解码输出缓冲区不小于 L_FRAME(160, 以 HI_S16 为单位)。

【举例】

```
#include "amr_dec.h"
HI_VOID *pDecState = NULL;
HI_U8 pInBuf[MAX_PACKED_SIZE];
HI_S16 pOutBuf[L_FRAME];
enum Format frame_type = MIME;
HI_S32 packed_size;

If (frame_type == MIME) /*本地文件解码时, 若帧格式为MIME, 则需要读出文件头*/
{
    fread(magic, sizeof(HI_S8), strlen(AMR_MAGIC_NUMBER), file_in);
    if (strncmp((const HI_S8 *)magic, AMR_MAGIC_NUMBER,
        strlen(AMR_MAGIC_NUMBER)))
    {
        fprintf(stderr, "%s%s\n", "Invalid magic number: ", magic);
        exit(-1);
    }
}
if (AMR_Decode_Init(&pDecState))
{
    fprintf(stderr, "\nerror AMR_Decode_Init fail: %s\n", strerror(errno));
    exit(-1);
}

while(fread (&pInBuf[0], sizeof(HI_U8), 1, file_in) == 1)
{
    packed_size = AMR_Get_Length(frame_type, pInBuf[0]);
    fread(&pInBuf[1], sizeof(HI_U8), packed_size, file_in);
    AMR_Decode_Frame(pDecState, pInBuf, pOutBuf, frame_type);
    fwrite (pOutBuf, sizeof (HI_S16), L_FRAME, file_out);
}

AMR_Decode_Exit(&pDecState);
```



【相关主题】

- HI_S32 AMR_Decode_Init
- HI_S32 AMR_Get_Length
- HI_VOID AMR_Decode_Exit



3 其他信息

3.1 数据类型

常数定义

```
/*帧长，编码器输入数据个数，解码器输出数据个数（HI_S16）。*/  
#define L_FRAME          160  
/*编码器打包后数据个数最大值，解码器输入解码数据个数最大值（HI_U8）。*/  
#define MAX_PACKED_SIZE   35  
/*采用MIME帧格式做本地存储时的文件头*/  
#define AMR_MAGIC_NUMBER  "#!AMR\n"
```

enum Mode

【说明】

AMR-NB 编解码器所支持的基本速率。

【定义】

```
enum Mode {  
    MR475 = 0,    /*可选速率，4.75 kbit/s    */  
    MR515,        /*可选速率，5.15 kbit/s    */  
    MR59,         /*可选速率，5.9 kbit/s     */  
    MR67,         /*可选速率，6.7 kbit/s     */  
    MR74,         /*可选速率，7.4 kbit/s     */  
    MR795,        /*可选速率，7.95 kbit/s    */  
    MR102,        /*可选速率，10.2 kbit/s    */  
    MR122,        /*可选速率，12.2 kbit/s    */  
    MRDTX,        /*静音模式，内部模式，不可选。*/  
    N_MODES       /*基本速率数目，不可选。    */  
}
```

**【注意】**

仅 MR475-MR122 等 8 种基本速率为用户可选。

enum Format**【说明】**

帧格式。定义了三种，分别为：

- MIME
RTP Payload format, 请参见 “rfc4239 AMR and AMR-WB Storage Format”。
- IF1
请参见 “3GPP 26101-600.doc”。
- IF2
请参见 “3GPP 26101-600.doc”。

【定义】

```
enum Format { MIME = 0, IF1, IF2 };
```

【注意】

采用 MIME 帧格式，本地存储为 .amr 文件，影音风暴支持播放。IF1 和 IF2 两种格式暂无播放器可播放。

3.2 错误码

错误码	描述
HI_ERR_AMRNB_INVALID_DEVICE	非法输入编解码设备指针。
HI_ERR_AMRNB_INVALID_INBUF	非法输入缓冲指针。
HI_ERR_AMRNB_INVALID_OUTBUF	非法输出缓冲指针。
HI_ERR_AMRNB_MODE_TYPE	非法编码速率。
HI_ERR_AMRNB_FORMAT_TYPE	非法帧格式。
HI_ERR_AMRNB_INIT_FAIL	编解码设备初始化失败。
HI_ERR_AMRNB_ENCODE_FAIL	编码失败。
HI_ERR_AMRNB_DECODE_FAIL	解码失败。



A 缩略语

Numerics

3GPP	3rd Generation Partnership Project	第三代移动通信标准化伙伴项目
-------------	------------------------------------	----------------

A

ACELP	Algebraic Code Excited Linear Prediction	代数码激励线性预测
--------------	--	-----------

AMR	Adaptive Multi-Rate	自适应多速率
------------	---------------------	--------

AMR-NB	Adaptive Multi-Rate Narrow-Band	窄带自适应多速率
---------------	---------------------------------	----------

C

CNG	Comfort Noise Generation	舒适噪声生成
------------	--------------------------	--------

CRC	Cyclic Redundancy Check	循环冗余检查
------------	-------------------------	--------

D

DTX	Discontinuous Transmission	非连续传输
------------	----------------------------	-------

F

FQI	Frame Quality Indication	帧质量描述符
------------	--------------------------	--------

I

IF1	AMR Interface Format 1	AMR 接口格式 1
------------	------------------------	------------

IF2	AMR Interface Format 2	AMR 接口格式 2
------------	------------------------	------------

M



A 缩略语

MMS	MIME file storage format	多用途的网际邮件扩充协议文件存储格式
P		
PCM	Pulse Code Modulation	调制脉冲编码
S		
SID	Silence Descriptor	静音描述符
T		
TS	Transport Stream	传输流
V		
VAD	Voice Activity Detection	语音检测