Witmem
知存科技

WTM2101

Hal audio 使用说明

版本号：V1.0.0.0
日期：2023.01.31

声明

商标声明：

Witmem 知存科技 作为北京知存科技有限公司的商标，本文件中提到的所有其他商标和商号均为其持有人的财产。

版权声明：

内容声明：

本文件中的信息如有更改，恕不另行通知。为了确保内容的准确性，文章会做出相关的确认，但本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

北京知存科技有限公司

地址：北京市海淀区北四环西路 56 号辉煌时代大厦西座 1502

网址：http://www.witintech.com

目录

一、 文档功能说明

该文档旨在说明 hal_audio 的驱动库.包括接口功能以及结构体参数意义.

二、 函数功能介绍

hal_audio_instance_get()

Audio_InitTypeDef* hal_audio_instance_get(Hal_Audio_Instance_Typedef number)
获取 hal audio 实例对象
参数
number：hal audio 实例对象编号
返回值
成功返回 Hal audio 实例对象,否则是 NULL

hal_audio_init()

int hal_audio_init(Audio_InitTypeDef *audio_instance,
Audio_Mic_Input_TypeDef type)
初始化 hal audio 实例对象
参数
audio_instance: hal audio 实例对象
type:audio 工作模式
返回值
大于 0 成功,否则失败
注意:初始化时根据 type 参数进行构造,可根据实际初始化后进行参数修改
当传入的 type 参数是 HAL_AUDIO_MIC_INPUT_AMIC,初始化 hal_audio 对象实例适配 AMIC,其他默认参数说明如下:
audio 前置分频系数 6,audio 分频系数 4
audio 使用通道 0
audio 通道数据位宽 16bits
audio 通道使用 ram buffer 存储数据
audio 通道增益 16
audio 通道采样数据在时钟上升沿
audio 通道使用 AnalogPreEmphasis 滤波器
audio 通道使用 sinc5d2 滤波器
audio 通道不使用 halfband 滤波器
audio 通道使用 highpass 滤波器

audio 通道不使用 PreEmphasis 滤波器
audio 通道 ram buffer 最大深度 480 个字
audio 通道 ram buffer 帧移大小 240 个字
audio 通道 ram buffer 中断响应大小 240 个字. 在使用 fifo buffer 时，此参数对于 audio 本身无意义. 但需要填充此数值,作为数据缓存大小. ex:获取 10ms 音频数据填 80,获取 30ms 音频数据填 240

audio 通道模拟 boostgain 是 20db
audio 通道模拟 pga 默认数值 15
audio 通道模拟 pga 最小数值 2
audio 通道模拟 pga 最大数值 20

audio 通道使能模拟 AGC
audio 通道模拟 AGC 的 OverSampling 16
audio 通道模拟 AGC 的 TargetThreshold (32768 * 0.1)
audio 通道模拟 AGC 的 MaximalThreshold (32768 * 0.2)
audio 通道模拟 AGC 的 ShortTermAttack -204
audio 通道模拟 AGC 的 LongTermAttack -10
audio 通道模拟 AGC 的 LongTermRelease 2

audio 通道使用 ram buffer 中断

audio 通道使用 dma 数据搬运
audio 通道使用 dma 通道 0
audio 默认关闭硬件 vad 功能

当传入的 type 参数是 HAL_AUDIO_MIC_INPUT_LINEIN 初始化 hal_audio 对象实例适配 LINEIN. 其他默认参数说明如下:

audio 前置分频系数 6,audio 分频系数 2
audio 使用通道 2
audio 通道数据位宽 16bits
audio 通道数据使用 fifo buffer
audio 通道增益 16
audio 通道采样数据在时钟上升沿
audio 通道使用 AnalogPreEmphasis 滤波器
audio 通道使用 sinc5d2 滤波器
audio 通道使用 halfband 滤波器

audio 通道使用 highpass 滤波器
通 audio 道不使用 PreEmphasis 滤波器
audio 通道 ram buffer 中断响应大小 240 个字.使用 fifo buffer 时，此参数对于 audio 本身无意义.但需要填充此数值,作为数据缓存大小. ex:获取 10ms 音频数据填 80,获取 30ms 音频数据填 240

audio 通道模拟 boostgain 是 18db
audio 通道模拟 pga 默认数值 15
audio 通道模拟 pga 最小数值 2
audio 通道模拟 pga 最大数值 20

audio 通道无模拟 AGC 功能

audio 通道使用 fifo buffer 中断

audio 通道不使用 dma 数据搬运
audio 默认关闭硬件 vad 功能

当传入的 type 参数是 HAL_AUDIO_MIC_INPUT_DMIC 初始化 hal_audio 对象实例适配 DMIC 其他默认参数说明如下:
audio 前置分频系数 6,audio 分频系数 4
audio 使用通道 0
audio 通道数据位宽 16bits
audio 通道关联 pdm0,对应 GPIO4,GPIO5
audio 通道数据使用 ram buffer
audio 通道增益 16
audio 通道采样数据在时钟上升沿
audio 通道不使用 AnalogPreEmphasis 滤波器
audio 通道使用 sinc5d2 滤波器
audio 通道不使用 halfband 滤波器
audio 通道使用 highpass 滤波器
audio 通道不使用 PreEmphasis 滤波器
audio 通道 ram buffer 最大深度 480 个字
audio 通道 ram buffer 帧移大小 240 个字
audio 通道 ram buffer 中断响应大小 240 个字.在使用 fifo buffer 时，此参数对于 audio 本身无意义. 但需要填充此数值,作为数据缓存大小. ex:获取 10ms 音频数据填 80,获取 30ms 音频数据填 240

audio 通道使用 ram buffer 中断

audio 通道使用 dma 数据搬运
audio 通道使用 dma 通道 0
audio 默认关闭硬件 vad 功能

hal_audio_open()

int hal_audio_open(Audio_InitTypeDef *audio_instance)
根据传入的 hal audio 实例对象打开相应的硬件
参数
audio_instance: hal audio 实例对象
返回值
大于 0 成功,否则失败

hal_audio_ctl()

int hal_audio_ctl(Audio_InitTypeDef *audio_instance,int command, ...)
控制 hal audio 实例对象或硬件参数
参数
command:命令参数
...:命令参数可能附加参数
返回值
大于 0 成功,否则失败
注意:
命令参数:
HAL_AUDIO_CHANNEL_ENABLE_COMMAND 表示开启与关闭 audio 的通道
HAL_AUDIO_INTERRUPT_ENABLE_COMMAND 表示使能与禁止 audio 通道中断
HAL_AUDIO_CHANNEL_DATA_WIDTH16BITS_COMMAND 表示切换 pdm 的 16bits 与 pcm 的 12bits 通道.当附加参数是 ENABLE,使能 pdm 16bits 通道
HAL_AUDIO_CLOCK_SWITCH_COMMAND 表示切换 audio 通道的时钟源
HAL_VAD_INTERRUPT_ENABLE_COMMAND 表示使能与禁止硬件 VAD

hal_audio_read()

int hal_audio_read(Audio_InitTypeDef *audio_instance,uint8_t *buffer)
读数据
参数
audio_instance:hal audio 实例对象
buffer:数据 buffer,使用 audio fifo buffer 有效
注意,如果使用 audio ram buffer,此 buffer 置 NULL.
返回值
大于 0 成功,否则失败

hal_audio_close()

int hal_audio_close(Audio_InitTypeDef *audio_instance)
关闭 hal audio 实例对象及关联的硬件
参数
audio_instance:hal audio 实例对象
返回值
大于 0 成功,否则失败

三、 结构体介绍

Audio_InitTypeDef

Hal audio 配置结构体

```
typedef struct{
    FunctionalState                 enable;
    AUD_TypeDef                     *instance;
    FunctionalState                 audio_rst;
    uint32_t                        prescale;
    uint32_t                        divider;
    Audio_Mic_Input_TypeDef         InputType;
    Audio_Connect_Pdm_Typedef       Pdm;
    Audio_Channel_InitTypeDef       channel;
    Audio_Analog_InitConfig         Analog;
    Audio_Interrupt_Config_TypeDef  interrupt;
    FunctionalState                 dma_enable;
    uint32_t                        dma_channel;
    volatile uint32_t               dma_finish_flag;
    DMA_InitTypeDef                 dma;
    FunctionalState                 vad_flag;
    HAL_AUDIO_CACHE                 audio_cache;
    Hal_Audio_Data_handle           Data_handle_info;
}Audio_InitTypeDef;
```

参数

enable:初始化完成标志

instance:audio 硬件地址

audio_rst:audio 硬件复位标志

prescale:audio 前置分频系数

divider:audio 模块分频系数

InputType:audio 工作类型

Pdm:audio 的 pdm 配置结构体

channel:audio 的通道配置结构体

Analog:audio 使用模拟 mic 时配置结构体

Interrupt:audio 的中断类型

dma_enable:启用 dma 的标志

dma_channel:dma 使用的通道
dma_finish_flag:dma 搬运数据时完成标志
dma:dma 配置结构体
vad_flag:启用硬件 vad 标志
Data_handle_info:内部数据处理接口
audio_cache:hal audio 数据 buffer 配置结构体

Audio_Connect_Pdm_Typedef

Audio pdm 配置结构体

```
typedef struct{
   FunctionalState     enable;
   Audio_Connect_Pdm_Number_TypeDef PdmNumber;
   uint32_t Pdm_Clk_Io,Pdm_Data_Io;
   uint32_t Pdm_Clk_IO_AF,Pdm_Data_Io_AF;
}Audio_Connect_Pdm_Typedef;
```

参数

enable:启用 audio pdm 标志
PdmNumber:audio 通道关联 pdm 编号
Pdm_Clk_Io,Pdm_Data_Io:audio 通道 pdm 使用的 gpio
Pdm_Clk_IO_AF,Pdm_Data_Io_AF:audio 通道 pdm 使用 gpio 功能选择

Audio_Channel_InitTypeDef

Audio channel 配置结构体

```
typedef struct{
   Audio_Channel_Number_TypeDef       ChannelNumber;
   Audio_Channel_Width_TypeDef        ChannelWidth;
   Audio_Buffer_Mode_TypeDef          BufferMode;
   Audio_Buffer_Date_Source_TypeDef   Source   ;
   uint32_t                           ChannelGain;
   Audio_Clk_Edge_Capture_TypeDef     Clk_Edge;
   Audio_Filter_Parameter_Typedef   Filter;
   uint32_t
```

Buffer_Ram_Depth,Buffer_Ram_Frame_Move,Buffer_Ram_Length
}Audio_Channel_InitTypeDef;
参数:
ChannelNumber:audio 使用的通道号
ChannelWidth:通道使用 pdm 16bits 还是 pcm 12bits
BufferMode:audio 通道数据存储类型
Source:audio 通道 buffer 数据来自数字还是模拟
ChannelGain:audio 通道增益
Clk_Edge:audio 通道时钟采样类型
Filter:audio 通道滤波配置结构体
Buffer_Ram_Depth: ram buffer 最大存储数据大小
Buffer_Ram_Frame_Move:ram buffer 数据帧移大小
Buffer_Ram_Length:ram buffer 触发中断数据大小

HAL_AUDIO_CACHE

Hal audio 数据缓存结构体

```
typedef struct{
   union{
      struct{
         volatile uint8_t    sram_data_flag;
         uint8_t             *sram_data;
      }sram;
      Ring_Cache             ring;
   }cache;
}HAL_AUDIO_CACHE;
```

参数
sram_data_flag:使用 audio ram buffer 时数据传输标志位
sram_data:使用 audio ram buffer 时数据缓存 buffer
ring:使用 audio fifo buffer 时使用环形缓冲区 buffer

四、 其他

1.audio 数据存储 buffer 申请自 heap.c 文件中的数组 static uint8_t ucHeap[configTOTAL_HEAP_SIZE] __attribute__ ((section(".audmem")));
在使用前需要自己在链接脚本中定义一块安全可用的地址,地址范围根据属性__attribute__ ((section(".audmem")))来指定.数组大小由宏 configTOTAL_HEAP_SIZE 定义

2.在 hal_audio.c 中的中断子程序只是示例代码,实际使用时务必重写中断子程序,否则会重现严重的错误.

五、 修订历史

表 5-1 修订历史

版本	日期	修订人	说明
V1.0.0.0	2023-01-31	李剑	初次编写