A76xx Series Open SDK _TTS_应用指导

LTE 模组

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
技术支持邮箱：support@simcom.com
官网：www.simcom.com

名称：	A76xx Series Open SDK_TTS_应用指导
版本：	V1.00
类别：	应用文档
状态：	已发布

版权声明

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
邮箱：simcom@simcom.com
官网：www.simcom.com

了解更多资料，请点击以下链接：
http://cn.simcom.com/download/list-230-cn.html

技术支持，请点击以下链接：
http://cn.simcom.com/ask/index-cn.html 或发送邮件至 support@simcom.com

Version History

Version	Date	Owner	What is new
V1.00	2022-11-22		第一版

About this Document

本文档适用于 A1803S open 系列、A1603 open 系列、A1606 open 系列。

目录

缩略语

TTS	Text To Speech
UNICODE	Universal Multiple-Octet Coded Character Set
ASCII	American Standard Code for Information Interchange
GBK	Chinese Internal Code Specification
UTF8	Unicode/UCS Transformation Format
MBCS	Multi-Byte Chactacter System (Set)
YT	优同
IFLY	科大讯飞

1 TTS 概述

TTS 是 Text To Speech 的缩写，即“从文本到语音”。它主要运用到的技术就是语音合成技术，而我们的模块主要是利用已有的语音数据库，把文字智能地转化为自然语音流。

TTS 语音合成技术已基本覆盖国标一、二级汉字，具有英文接口，自动识别中、英文，支持中英文混读。

下面将通过多方面介绍 TTS (语音合成技术)：

1.1 TTS 文本转语音流程

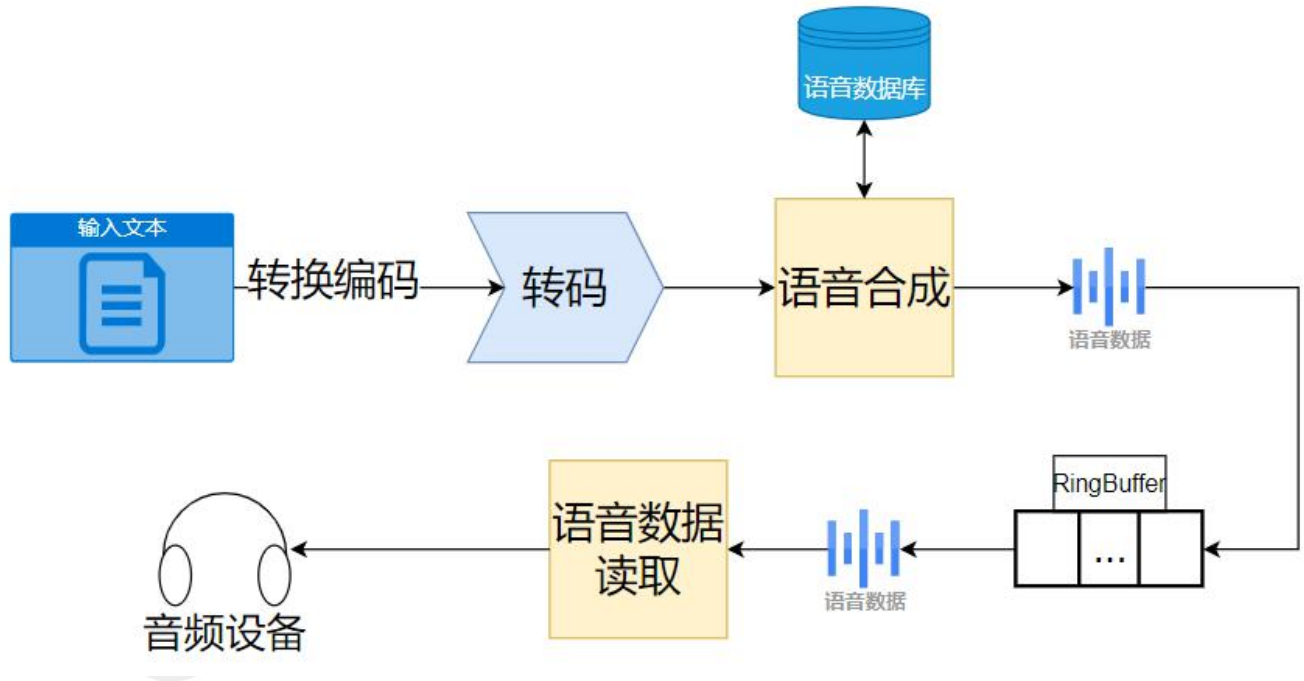

图 2.1 文本转语音流程图

1.2 TTS 底层实现描述（YT 为例）

1) 通过 **sAPI_TTSPlay()** 接口输入 Text 文本；

2) 对输入的 Text 文本信息进行编码转换（可支持 UNICODE，ASCII or GBK 码）；

3) 调用 YT_TTS_SynthesisThreadFunc() 合成线程函数进行语音合成；

4) 将合成后的语音数据写入 RingBuffer （缓冲区的大小可以根据资源情况进行设置）；

5) 将合成的语音数据从 RingBuffer 中每次取一小块语音数据（512 个样点，具体数值可以更改）；

6) 调用 YT_TTS_PlayThreadFunc() 播放线程函数，将语音数据写入到音频设备进行语音播放。

可实现边合成边播放，响应效率大大提高，合成可以根据上层需要随时停止、继续，最小响应时间可以达到 20ms。

2TTS 功能特性

1） 功能特性简要描述如下：

编码格式：

- 支持 UTF8/UTF16/MBCS 等多种编码；

支持语言：

- IFLY TTS 语音库默认只支持中文，英文或者中英文混读需要定制版本支持；
- YT TTS 语音库支持中文、英文、中英文混读；

其他功能：

- 可实现特殊符号和数字的智能处理；
- 可自由调节音高和语速；
- 可合成清晰流畅的语音；
- 能模仿真人表现出特定的语气。

2） 语音合成后的数据格式如下：

采样率：

- 支持 16000Hz 、8000Hz；

量化位数：

- 使用 16 bit, 线性 PCM

声道数：

- 单声道

3TTS API 介绍

头文件：#include “simcom_tts_api.h”

3.1 sAPI_TTSPlay

此应用程序接口用于播放 **TTS** 文本。

接口:	BOOL sAPI_TTSPlay(UINT8 option, char *inputText, UINT8 playMode);
输入:	option: 设置编码格式； 1 UNICODE（UCS）编码 2 ASCII or GBK 编码 inputText： 输入需要播放的文本；最大长度为 508 字节。请将较长的英文文本分割，否则导致不能完整地阅读数据； playMode：播放路径； 0 本地播放 1 远程播放；
输出:	无
返回值:	True: 播放成功 False: 播放失败
NOTE:	无

3.2 sAPI_TTSStop

此应用程序接口用于停止 **TTS** 播放。

接口:	BOOL **sAPI_TTSStop**(void);
输入:	无
输出:	无
返回值:	True: 停止播放成功 False: 停止播放失败
NOTE:	无

3.3 sAPI_TTSSetParameters

此应用程序接口用于设置 **TTS** 参数。

接口:	BOOL sAPI_TTSSetParameters(UINT8 volume, UINT8 sysVolume, UINT8 digitMode, UINT8 pitch, UINT8 speed);
输入:	volume: 模块系统音量，取值范围：0 ~ 11，默认值：5（YT、IFLY 取值范围相同） sysVolume: TTS 库音量 YT: 取值范围：0 ~ 3，默认值：3（最大音量） IFLY：取值范围：0 ~ 2，默认值：2（最大音量） digitMode: 数字读取模式，取值范围：0 ~ 3，默认值：0 YT： 0 是默认读取模式；1 是电话号码读取模式；2 是数字读取模式；3 是值读取模式（比如 123 读作一百二十三） IFLY: 0 是默认读取模式；1 是数字读取模式；2 是值读取模式；3 是数字读取模式 pitch: 语音音高，取值范围：0 ~ 2，默认值：1 speed: 语音速度，取值范围：0 ~ 2，默认值：1
输出:	无
返回值:	True: 设置参数成功 False: 设置参数失败
NOTE:	无

3.4 sAPI_TTSSetStatusCallBack

此应用程序接口用来设置 **TTS** 状态回调函数。

接口:	BOOL sAPI_TTSSetStatusCallBack(sAPI_TTSStatussCb ttsCb);
输入:	ttsCb: 注册用于报告 TTS 状态的回调函数
输出:	无
返回值:	True: 注册成功 False: 注册失败
NOTE:	无

4TTS 数据定义

4.1 TTSPlayerStatus

```
typedef enum
 {
     NO_WORKING,        //TTS 未工作
     TTS_WORKING,       // TTS 正在工作
     STATE_MAX,
}TTSPlayerStatus;        // TTS 播放状态
```

4.2 TTSStatus

```
typedef enum
{
     SC_TTS_STOP,          //TTS 停止运行
     SC_TTS_START,         //TTS 开始运行
     SC_TTS_FAIL,          //TTS 语音合成失败
     SC_TTS_STATUS_MAX,
}TTSStatus;        // TTS 工作状态
```

5TTS Demo

```
#include “simcom_tts_api.h”

void TTSDemo(void)
{
    SIM_MSG_T optionMsg = { 0 };
    char text[512] = { 0 };
    UINT32 opt, option, mode, volume, tts_volume, digit_mode, pitch, speed = 0;
    BOOL ret = FALSE;
    char *note = "\r\nPlease select an option to test from the items listed below.\r\n";
    char *options_list[] ={
        "1. Play",
        "2. Stop",
        "3. SetPara",
        "99. back",
    };

    while(1)
    {
        PrintfResp(note);
        PrintfOptionMenu(options_list,sizeof(options_list)/sizeof(options_list[0]));
        sAPI_MsgQRecv(simcomUI_msgq, &optionMsg, SC_SUSPEND);
        if (SRV_UART != optionMsg.msg_id)
        {
            sAPI_Debug("%s, msg_id is error!!!", __func__);
            break;
        }

        sAPI_Debug("arg3 = [%s]", optionMsg.arg3);
        opt = atoi(optionMsg.arg3);
        sAPI_Free(optionMsg.arg3);

        switch(opt)
        {
            case SC_TTS_DEMO_PLAY:
            {
                PrintfResp("\r\nPlease option.(1.UNICODE,2.ASCII)\r\n");
                optionMsg = GetParamFromUart();
                option = atoi(optionMsg.arg3);
                sAPI_Free(optionMsg.arg3);
```

```
        sAPI_Debug("sAPI_TTSPlay: option is %d ",option);

        PrintfResp("\r\nPlease input TTS text.\r\n");
        optionMsg = GetParamFromUart();
        strcpy(text,optionMsg.arg3);
        sAPI_Free(optionMsg.arg3);
        sAPI_Debug("sAPI_TTSPlay: text is %s ",text);

        PrintfResp("\r\nPlease input play mode.(0.local,1.remote)\r\n");
        optionMsg = GetParamFromUart();
        mode = atoi(optionMsg.arg3);
        sAPI_Free(optionMsg.arg3);
        sAPI_Debug("sAPI_TTSPlay: playMode is %d ",mode);

        ret = sAPI_TTSPlay(option,text, mode);
        if(ret)
            sAPI_Debug("sAPI_TTSPlay: playing    success ");
        else
            sAPI_Debug("sAPI_TTSPlay: playing    failed");
        break;
    }

    case SC_TTS_DEMO_STOP:
    {
        ret = sAPI_TTSStop();
        if(ret)
            sAPI_Debug("sAPI_TTSStop: stop success ");
        else
            sAPI_Debug("sAPI_TTSStop: stop failed");
        break;
    }

    case SC_TTS_DEMO_PARA:
    {
        PrintfResp("\r\nPlease input system volme.\r\n");
        optionMsg = GetParamFromUart();
        volume = atoi(optionMsg.arg3);
        sAPI_Free(optionMsg.arg3);

        PrintfResp("\r\nPlease input tts volume.\r\n");
        optionMsg = GetParamFromUart();
        tts_volume = atoi(optionMsg.arg3);
        sAPI_Free(optionMsg.arg3);

        PrintfResp("\r\nPlease input digit mode.\r\n");
        optionMsg = GetParamFromUart();
```

```
                digit_mode = atoi(optionMsg.arg3);
                sAPI_Free(optionMsg.arg3);

                PrintfResp("\r\nPlease input pitch.\r\n");
                optionMsg = GetParamFromUart();
                pitch = atoi(optionMsg.arg3);
                sAPI_Free(optionMsg.arg3);

                PrintfResp("\r\nPlease input speed.\r\n");
                optionMsg = GetParamFromUart();
                speed = atoi(optionMsg.arg3);
                sAPI_Free(optionMsg.arg3);

                ret = sAPI_TTSSetParameters(volume, tts_volume, digit_mode,pitch,speed);
                if(ret){
                    PrintfResp("\r\nSet parameters success\r\n");
                    sAPI_Debug("sAPI_TTSSetParameters: set success");
                }
                else{
                    PrintfResp("\r\nSet parameters faild\r\n");
                    sAPI_Debug("sAPI_TTSSetParameters: set failed");
                }
                break;
            }

            case SC_AUDIO_DEMO_MAX:
            {
                return;
            }
        }
    }
}
```