Rockchip Android14 Media Streaming AI Subtitle Application Function Description

ID: RK-SM-YF-F14

Release Version: V1.0.0

Release Date: 2025-08-04

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document is the functional specification for AI automatic subtitles development on RK3576 Android 14.0 Media Streaming platform.

Product Version

Chipset	Android Version
RK3576	Android14.0

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	yi.yang	2025-08-04	Initial version

Contents

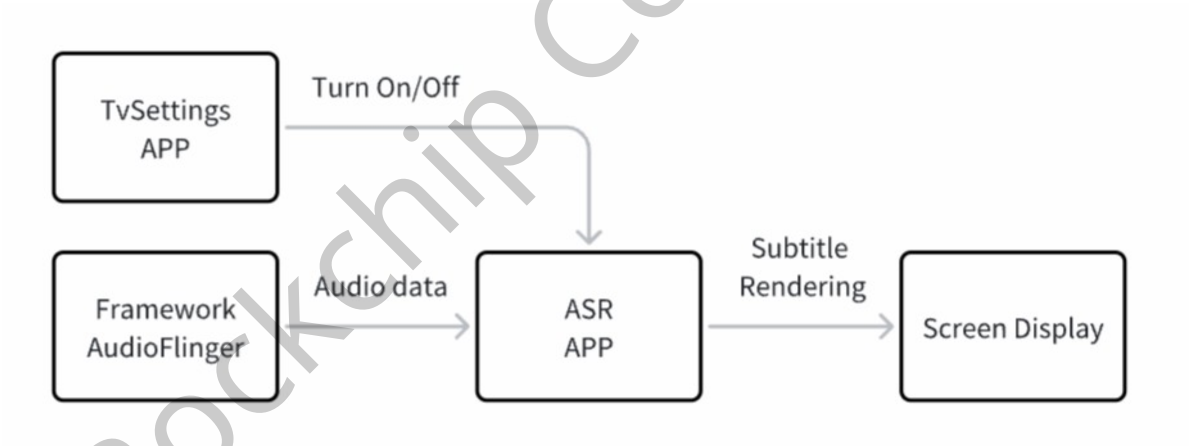
1. Overview

AI Auto Caption is a real-time speech-to-text system based on the Android audio framework. Its main function is to perform speech recognition through audio input (microphone or system audio) and display the recognition results as subtitles on the screen. The system supports bilingual recognition (Chinese and English) and features low latency and high accuracy. It is specifically designed with a **Meeting Mode** for video conferencing scenarios.

2. Overall Functional Framework

The AI caption function can be divided into three main modules:

1. **Audio Data Acquisition** (framework/av): Through the `AudioFlinger` framework at the Framework layer, obtain audio data from input devices (microphone) or system audio output (e.g., media playback streams), and transmit the audio data to the ASR module APP using the `IRkAiCallback` interface.
2. **Audio Recognition and Subtitle Display** (RkBoxAiService): The AI subtitle APP receives audio data from the Framework layer, invokes the ROCKASR interface of the NPU, sends the audio data to the underlying layer, retrieves the string recognition results from the underlying layer, and renders the subtitles for display.
3. **UI Settings Interface** (TvSettings): Used to enable the AI subtitle feature in the settings interface, currently offering three modes: Default Source, Chinese Source, and Meeting Mode.



Features include:

1. **Dual-Channel Speech Recognition**

- AT Path (System Audio): Captures audio played by the system (e.g., video sound, remote call audio)
    - AR Path (Microphone Audio): Captures microphone-recorded audio
    - Conference Mode: Dual-path independent processing, supports parallel recognition, and independent subtitle display for dual channels

2. **Real-Time Subtitle Rendering**

- Floating window for subtitle display (supports dual-line display for original/translated text)
- Timeout management: Automatically hides subtitles after 5 seconds of no new speech

3. **Multilingual Support**

- Mixed Chinese-English recognition capability
- Multiple modes can be switched arbitrarily (controlled via system properties)

3. Technical Implementation Solution

3.1 Framework Layer

Both system audio and recorded audio data sources are acquired in Threads.cpp and transmitted to the AI Subtitle APP side. The involved threads are shown in the following table:

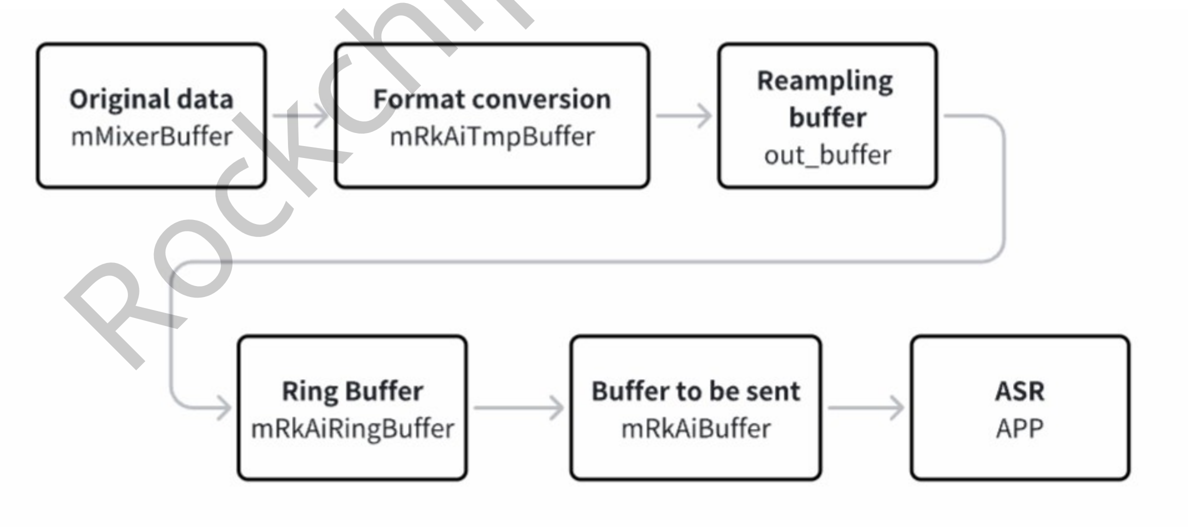
Audio Data	Processing Thread
System Audio	PlaybackThread
Recording Audio	RecordThread

3.1.1 Audio Data Collection

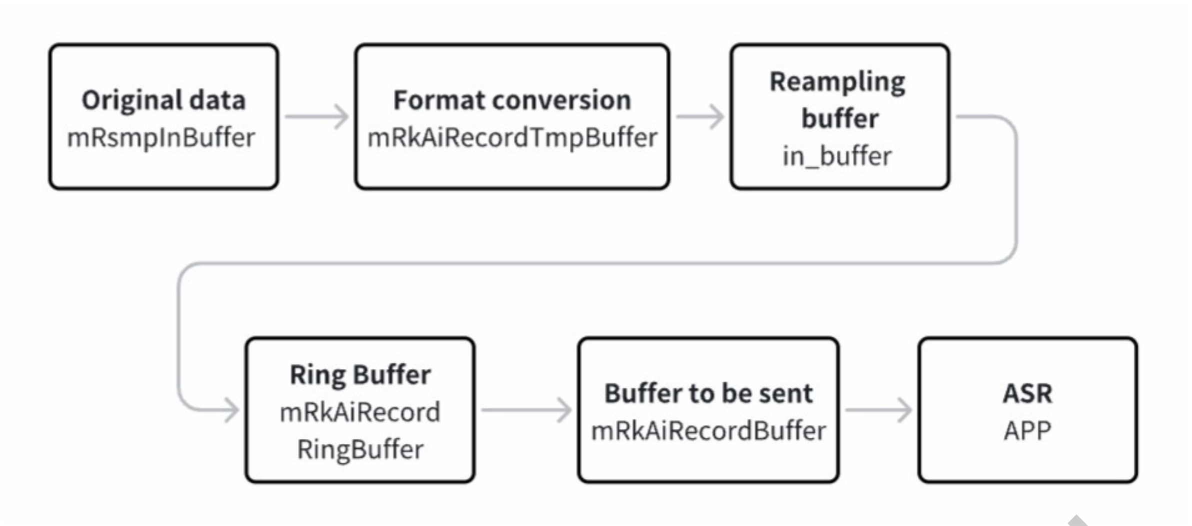
Whether it is system audio or recording audio, the process of obtaining audio data is the same:

> 1. Convert raw data to PCM audio format in uint16_t
> 2. Resample to the 16KHz sampling rate required by the underlying ROCKASR
> 3. Pass into the RK customized ring buffer, preparing for asynchronous audio data transmission
> 4. When the data fills one frame, send one frame of data to the AI subtitle APP side through the callback interface

The system audio data flow in the PlaybackThread is shown in the following figure



The recording audio data flow in the RecordThread is shown in the following figure



3.1.2 Audio Data Transmission

The callback interface `onAsrBuffer()` transmitted to the AI subtitle APP has been modified by adding the parameter `int isAudioRecord` to distinguish between system/recording audio data.

```
// media/libaudioclient/aidl/android/media/IRkAiCallback.aidl
/* Interface used to send pcm data from audioflinger to asrmanager */
interface IRkAiCallback {
   oneway void onAsrBuffer(in int[] buffer, int len, int isAudioRecord);
}
```

The definition corresponding to `int isAudioRecord` is as follows

```
#define ASR_AT_CHANNEL_INPUT 0 //System audio data
#define ASR_AR_CHANNEL_INPUT 1 //Recording audio data
```

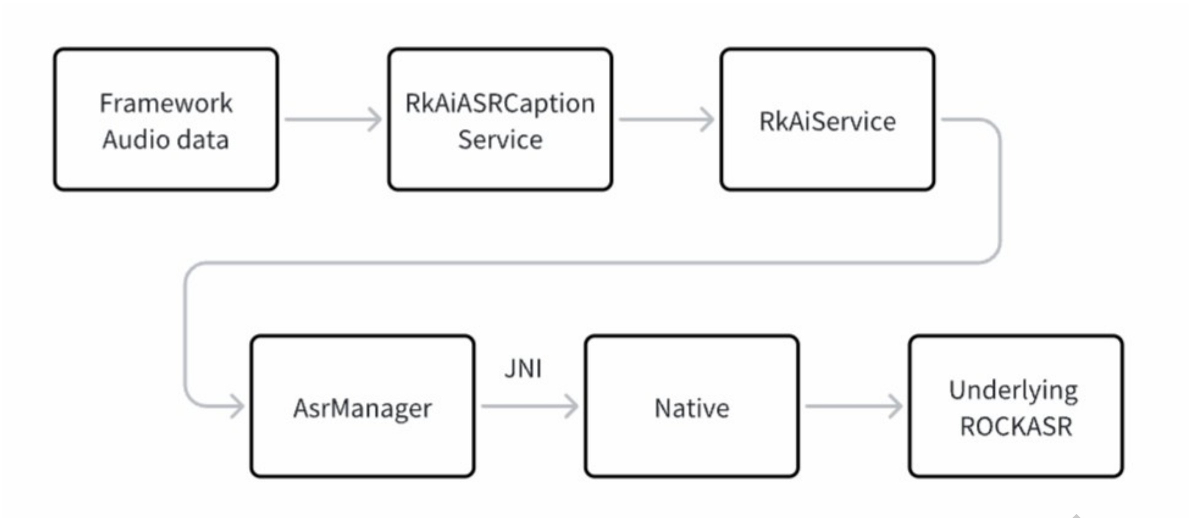
3.2 RkBoxAiService Module

Specifically the AI subtitle APP side, mainly containing two services

Service	Function
RkAiASRCaptionService	Manages audio data streams Controls subtitle display components
RkAiService	Manages communication with Native layer

3.2.1 Audio Data Transmission

The acquired audio data is passed down to the underlying ROCKASR for AI speech recognition processing. The audio data flow is shown in the following diagram:



The relevant interfaces all use `int isAudioRecord` to distinguish between system/recording audio data.

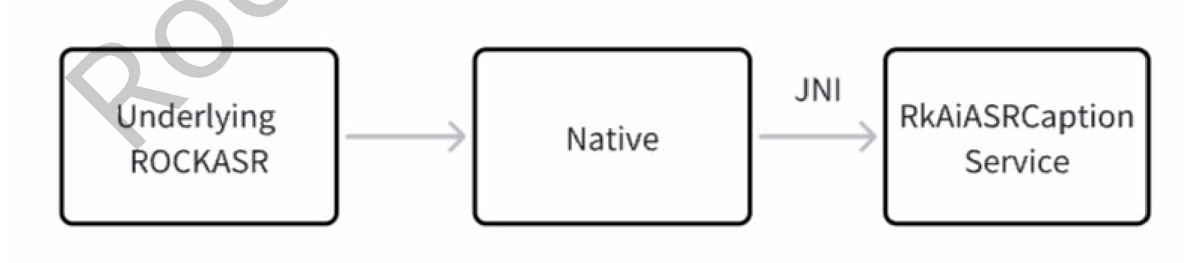
```
// aidl/com/rockchip/aiservice/IRkAiService.aidl
boolean requestAsr(String packageName, in int[] inputBuff, int buffSize, int
isAudioRecord);
void flushAsrInput(String packageName, int isAudioRecord);

// src/com/rockchip/aiservice/util/JniTool.java
public native int native_asr_inference(short[] buffer, int readsize, int
isAudioRecord);
public native int native_asr_flush(int isAudioRecord);
```

```
// cpp/RkAiServiceJni.cpp
JNIEXPORT jint JNICALL native_asr_inference(
        JNIEnv *env, jobject thiz, jshortArray data, jint dataSize, jint
isAudioRecord)
```

3.2.2 Recognition Result Transmission

After AI recognition, the ROCKASR returns a string result, which is directly passed to RkAiASRCaptionService through a callback and ultimately displayed in the captions. The string transfer is illustrated below.



The related interfaces all use `ASR_CALL_STATE state` to determine the recognition result corresponding to the returned string.

```
// src/com/rockchip/aiservice/util/JniTool.java
public interface MyAsrCallback {
    void asrInitCallBack(int ret);
    void asrCallBack(String result, ASR_CALL_STATE state);
}
```

```
// src/com/rockchip/aiservice/service/RkAiASRCaptionService.java
public void asrCallBack(String result, ASR_CALL_STATE state) {
    ……
}
```

`ASR_CALL_STATE state` has four states, defined as follows

```
//cpp/RkAiServiceJni.h
#define AT_SOURCE 0 //System audio original result
#define AT_RESULT 1 //System audio translation result
#define AR_SOURCE 2 //Recorded audio original result
#define AR_RESULT 3 //Recorded audio translation result

//src/com/rockchip/aiservice/service/RkAiASRCaptionService.java
private final int AT_SOURCE = 0; //System audio original result
private final int AT_RESULT = 1; //System audio translation result
private final int AR_SOURCE = 2; //Recorded audio original result
private final int AR_RESULT = 3; //Recorded audio translation result
```

In RkAiASRCaptionService, use `ASR_CALL_STATE state` to determine which Subtitle's Textview should display the returned string. The actual parameter passed to `updateFloatingView()` is `int whichSubtitle`.

```
 private void updateFloatingView(String result, int whichSubtitle) {
    ……
    switch(whichSubtitle) {
        case AT_SOURCE : {
            ……
        } case AT_RESULT : {
            ……
        } case AR_SOURCE : {
            ……
        } case AR_RESULT : {
            ……
        } default :
            break;
    }
     ……
 }
```

3.3 TvSettings Module

For AI subtitle applications, there are four control options

```
 <string-array name="ai_lab_asr_choose">
        <item>@string/ai_lab_asr_close</item>
        <item>@string/ai_lab_asr_en</item>
        <item>@string/ai_lab_asr_cn</item>
        <item>@string/ai_lab_asr_meeting</item>
 </string-array>
```

The corresponding English characters are as follows

```
<string name="ai_lab_asr_close">close</string>
<string name="ai_lab_asr_en">default input</string>
<string name="ai_lab_asr_cn">chinese input</string>
<string name="ai_lab_asr_meeting">meeting mode</string>
```

The corresponding Simplified Chinese characters are as follows

```
<string name="ai_lab_asr_close">关闭</string>
<string name="ai_lab_asr_en">开启: 默认源</string>
<string name="ai_lab_asr_cn">开启: 中文源</string>
<string name="ai_lab_asr_meeting">开启: 会议模式</string>
```

When the user selects an option through the UI, it essentially modifies the following PROP attributes and then restarts the AI subtitle APP

```
private final String PROP_ASR_ENABLE_STATE = "sys.rkai.asr.enable";
private final String PROP_SUBTITLE_ENABLE_STATE = "sys.rkai.subtitle.enable";
private final String PROP_ASR_CHINESE_ENABLE_STATE = "sys.rkai.asr_ch.enable";
private final String PROP_ASR_MEETING_MODE_ENABLE_STATE =
"sys.rkai.asr_meeting.enable";
```

Among them, `PROP_ASR_CHINESE_ENABLE_STATE` and `PROP_ASR_MEETING_MODE_ENABLE_STATE` are attributes that directly control the modes. The meanings of each attribute are as follows:

No.	Property	Description
1	PROP_ASR_ENABLE_STATE (`sys.rkai.asr.enable`)	"0": Disable AI auto-captioning function "1": Enable AI auto-captioning function
2	PROP_SUBTITLE_ENABLE_STATE (`sys.rkai.subtitle.enable`)	Currently serves the same purpose, values remain synchronized
3	PROP_ASR_CHINESE_ENABLE_STATE (`sys.rkai.asr_ch.enable`)	"0": With translation mode, supports English->Chinese "1": No translation mode, supports mixed English/Chinese recognition
4	PROP_ASR_MEETING_MODE_ENABLE_STATE (`sys.rkai.asr_meeting.enable`)	"0": Disable meeting mode "1": Enable meeting mode, currently bound to no-translation mode

The values for each mode are as follows:

Mode (`Simplified Chinese`)	Attribute1	Attribute2	Attribute3	Attribute4
close	0	0	/	/
default input	1	1	0	0
chinese input	1	1	1	0
meeting mode	1	1	1	1

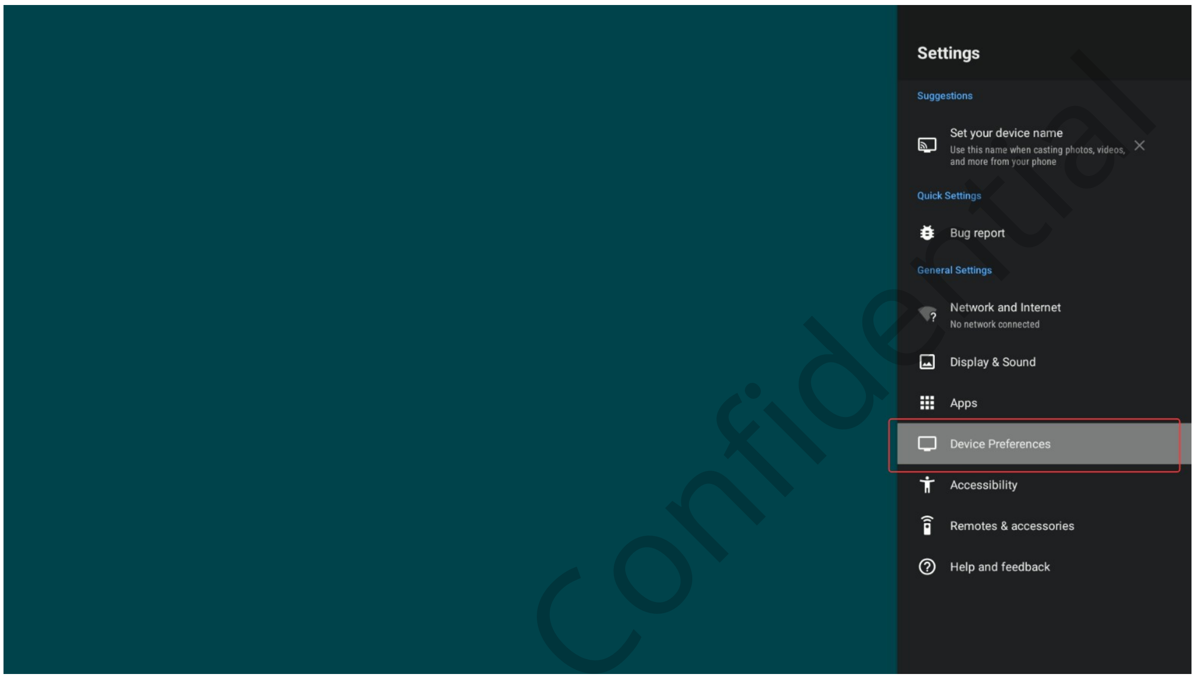
4. Usage Instructions

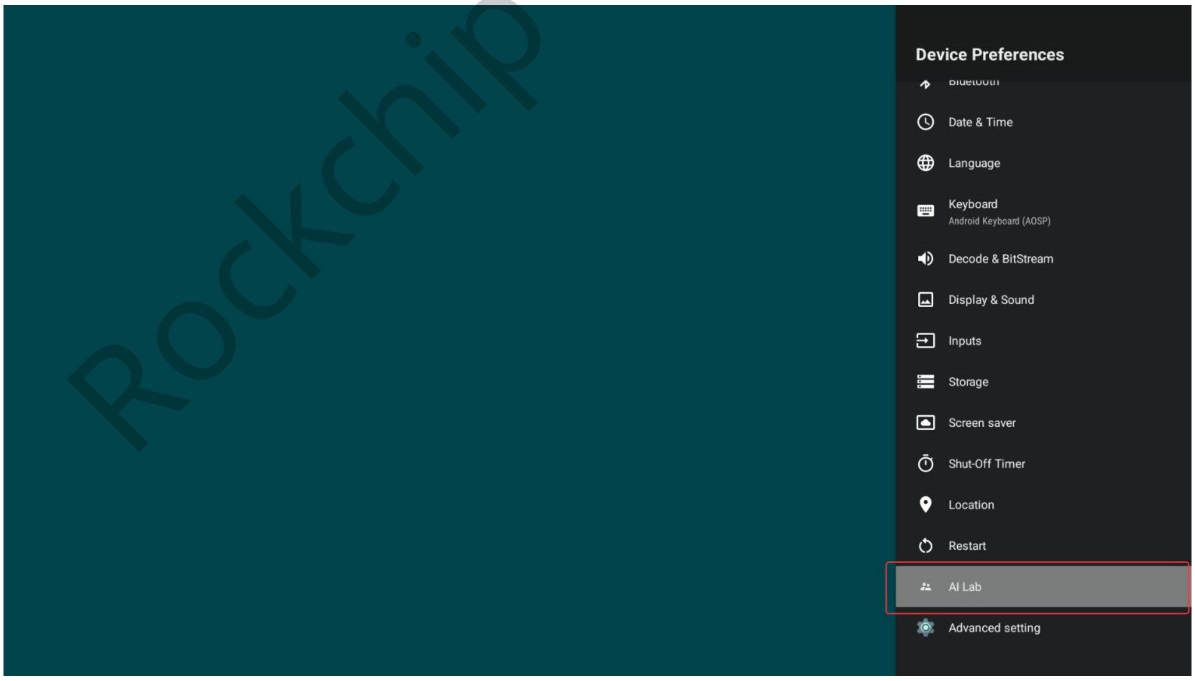
4.1 On/Off Settings

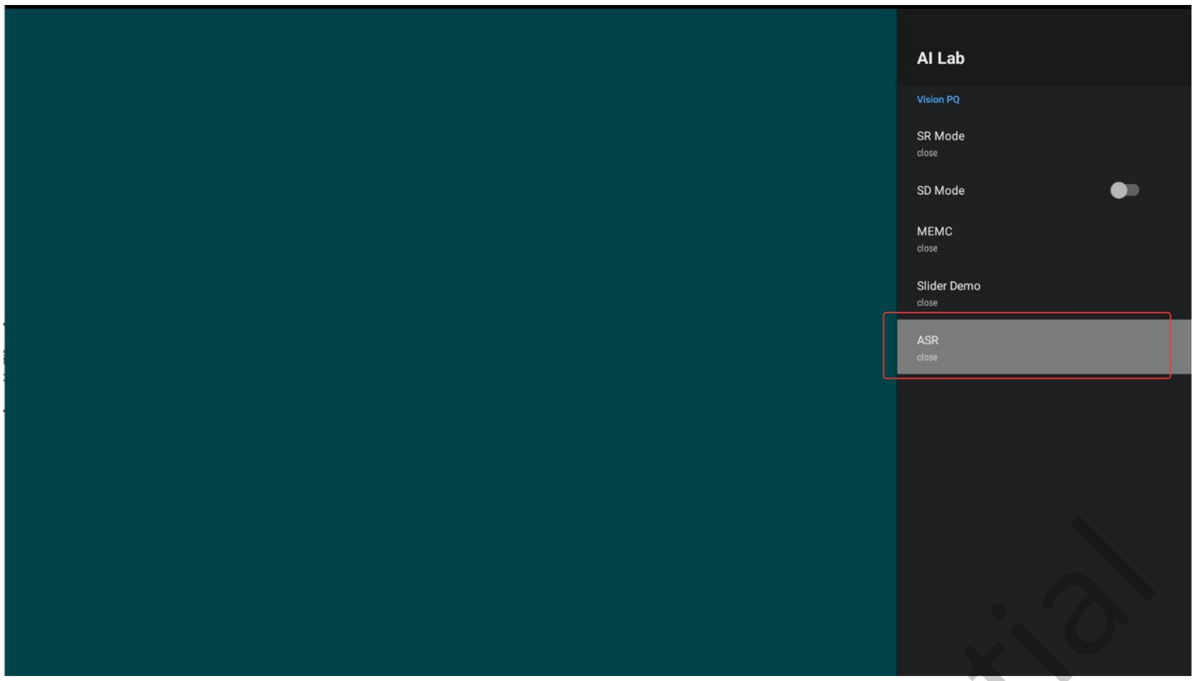
The AI auto-captioning feature needs to be manually enabled in the settings interface. If the system's `language` is set to `Simplified Chinese`, the navigation path is as follows:

`Settings -> Device Preferences -> AI Lab -> AI Auto-Captioning`

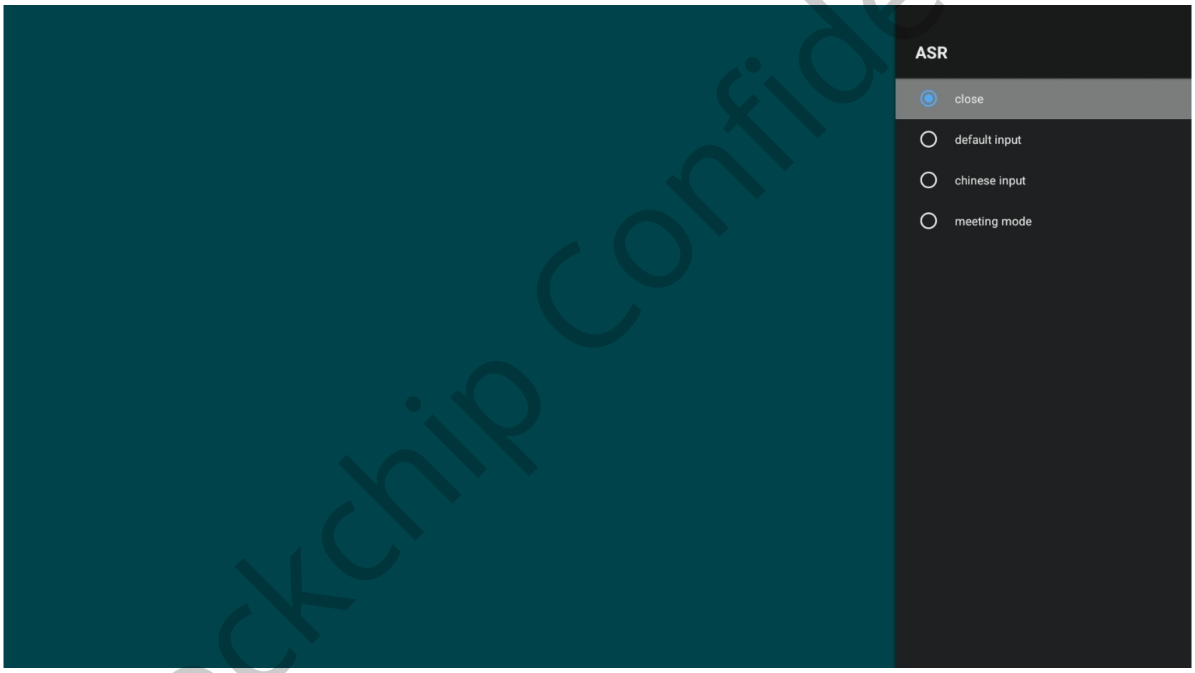
The same applies to other system `languages`







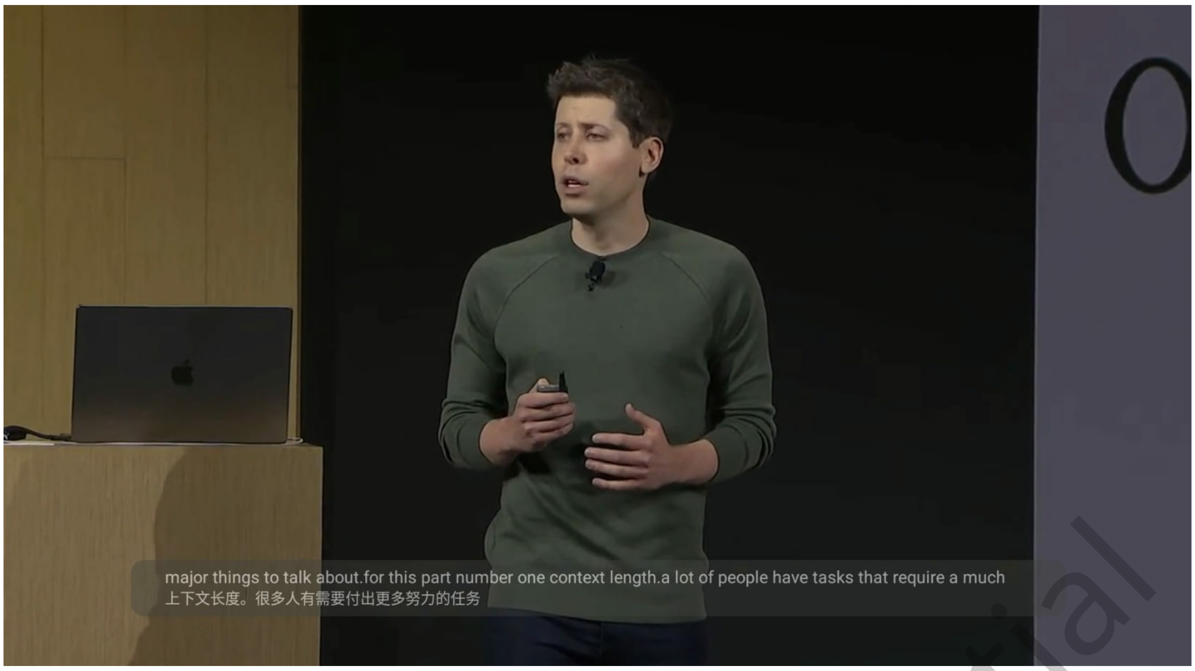
Enter the AI automatic subtitle settings interface, you can see there are currently four options, corresponding to the content in section 3.3



4.2 Effect Demonstration

4.2.1 Enable: Default Source (`ai_lab_asr_en`)

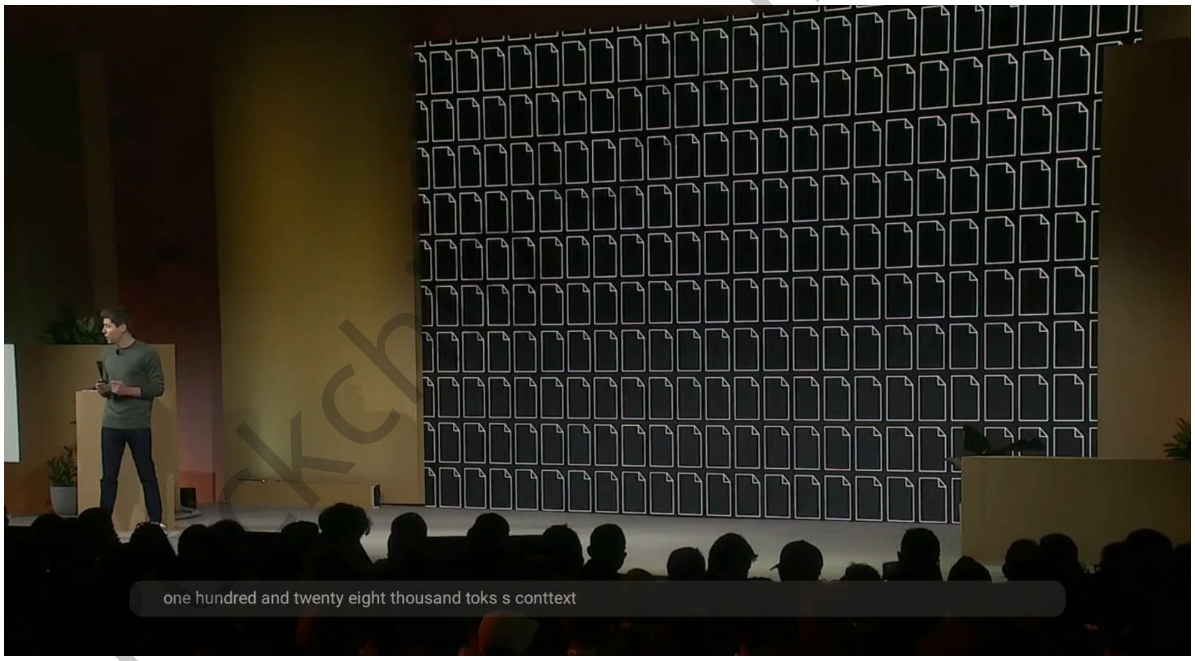
In this mode, only system audio data is captured and translated, displaying both the original text and the translation result



major things to talk about.for this part number one context length.a lot of people have tasks that require a much
上下文长度。很多人有需要付出更多努力的任务

4.2.2 Enable: Chinese Source (`ai_lab_asr_cn`)

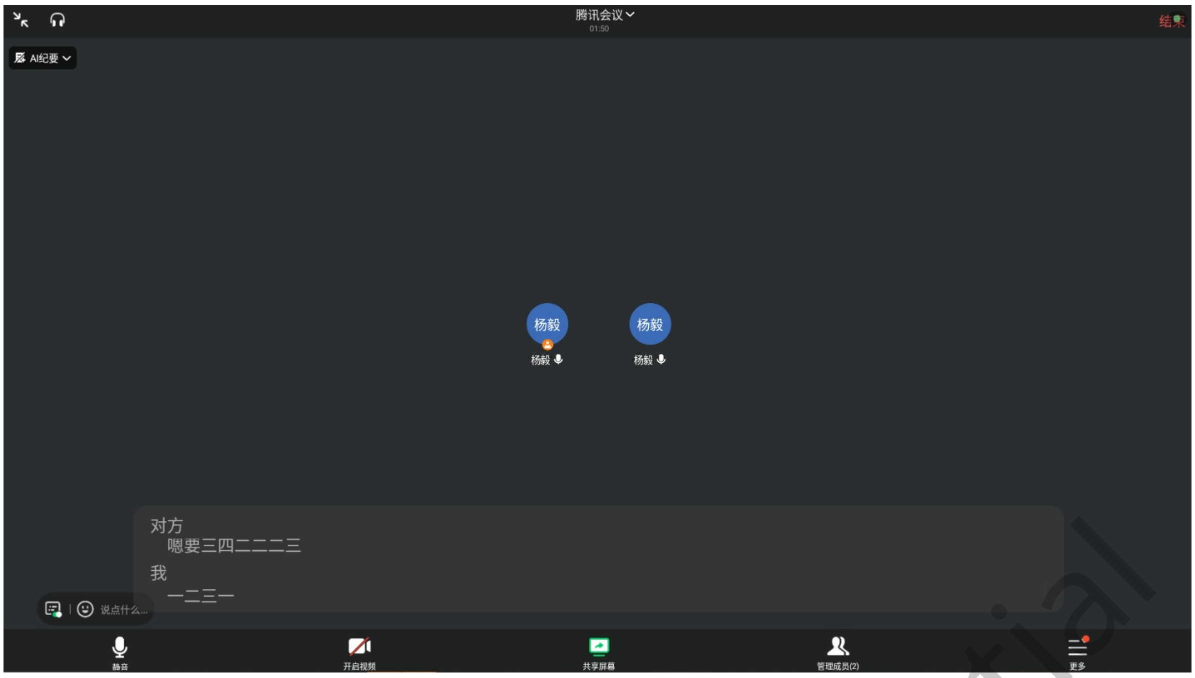
In this mode, only system audio data is captured without translation, displaying only the original recognition results with bilingual Chinese/English recognition.



one hundred and twenty eight thousand toks s conttext

4.2.3 Enable: Meeting Mode (`ai_lab_asr_meeting`)

This mode captures both system audio and recorded audio data without translation, supporting simultaneous display of original text results.



对方
嗯要三四二二二三
我
一二三一

5. FAQ

5.1 Debugging Single Module: Output Path and Android Device Storage Path

This feature involves multiple modules, each with different paths. Below are the commonly used paths for debugging this feature:

Module	**Output Path (AOSP Directory)**	**Download Path (Device Directory)**
RkBoxAiService Module	out/target/product/rk3576_projector/system/app/RkBoxAiService/	system/app/
RkBoxAiService Module	out/target/product/rk3576_projector/system/lib64/librkai_service.so	system/lib64
TvSettings Module	out/target/product/rk3576_projector/system_ext/priv-app/TvSettings/	system_ext/priv-app

5.2 Troubleshooting: No Subtitles Displayed After Enabling AI Caption Feature

1. Check if the system volume is set to 0. No subtitle output will be generated when muted as system audio data cannot be captured.
2. Verify rkauth authorization by checking if the `key_asr.lic` file exists under `/sdcard/rkai` via adb

```
ls -l /sdcard/rkai
```

Otherwise, rkauth authorization is required

```
//packages/apps/RkAuth/etc/rkauth_config.json
{
    "rkauth_modules_config": [
        ......
        {
            "module": "asr",//Module name: asr speech-to-text
            "activate_code_vendor_storage_id": 84,//ASR online key is stored at
vendorstorage position 84. Do not modify this ID.
            "activate_code_vendor_storage_size": 256,
            "license_vendor_storage_id": 1084,
            "license_vendor_storage_size": 1024,
            "activate_code_path": "",
            "license_path": "/sdcard/rkai/key_asr.lic" //RkAiService will read
this file for ASR initialization
        },
        ......
    ]
}
```

3. Check whether the device has stored the AI algorithm data files related to subtitle functionality. If not, they need to be reimported.

```
adb shell "ls /sdcard/rkai/asr_data"
```

4. Check whether the relevant macros are enabled in BoardConfig.mk under device/rockchip/rk3576/rk3576_projector when compiling the image

```
 #enable AIPQ
-BOARD_ENABLE_AIPQ_PROJECTOR_PRODUCT := false
+BOARD_ENABLE_AIPQ_PROJECTOR_PRODUCT := true

 BOARD_SELINUX_ENFORCING := false
@@ -61,9 +61,11 @@ BOARD_HDR_SUPPORT := true
 BOARD_SUPPORT_KEYSTONE := false

 ###support asr
-BOARD_SUPPORT_ASR := false
+BOARD_SUPPORT_ASR := true
```

5. Try to manually set `sys.rkai.subtitle.timeout`

```
adb shell "setprop sys.rkai.subtitle.timeout 5000" #unit ms, subtitle timeout
automatic hiding time
```

If it still doesn't work, check whether the prop property values comply with the description in section 3.3

```
adb shell "getprop | grep sys.rkai"
```

6. If none of the above resolves the issue, use `adb logcat` to capture logs for troubleshooting