



# **Linux Media**

## **开发指南**

**版本号: 1.0**  
**发布日期: 2021.04.13**

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.13	AWA1016	初始化 tplayer 相关接口使用说明以及 tplayerdemo 测试用例的使用说明



# 目 录

<b>1 概述</b>	<b>1</b>
1.1 编写目的 . . . . .	1
1.2 适用范围 . . . . .	1
1.3 相关人员 . . . . .	1
<b>2 软件环境配置</b>	<b>2</b>
2.1 内核配置 . . . . .	2
2.1.1 选中 ve 模块 . . . . .	2
2.1.2 选中 ion 模块 . . . . .	3
2.2 tina 配置 . . . . .	3
2.2.1 选中 tplayer 播放中间件 . . . . .	3
2.2.2 选中 tplayerdemo . . . . .	4
<b>3 TPlayer 状态图及状态说明</b>	<b>6</b>
3.1 TPlayer 状态图 . . . . .	6
3.2 TPlayer 每个状态简要说明 . . . . .	7
3.2.1 Idle 状态 . . . . .	7
3.2.2 Initialized 状态 . . . . .	7
3.2.3 Preparing 状态 . . . . .	7
3.2.4 Prepared 状态 . . . . .	7
3.2.5 Started 状态 . . . . .	7
3.2.6 Paused 状态 . . . . .	8
3.2.7 Stopped 状态 . . . . .	8
3.2.8 PlaybackCompleted 状态 . . . . .	8
3.2.9 Error 状态 . . . . .	8
3.2.10 End 状态 . . . . .	8
<b>4 接口函数说明</b>	<b>9</b>
4.1 TPlayerCreate . . . . .	9
4.2 TPlayerDestroy . . . . .	9
4.3 TPlayerSetDebugFlag . . . . .	9
4.4 TPlayerSetNotifyCallback . . . . .	10
4.5 TPlayerSetDataSource . . . . .	10
4.6 TPlayerPrepare . . . . .	10
4.7 TPlayerPrepareAsync . . . . .	10
4.8 TPlayerStart . . . . .	11
4.9 TPlayerPause . . . . .	11
4.10 TPlayerStop . . . . .	11
4.11 TPlayerReset . . . . .	11
4.12 TPlayerSeekTo . . . . .	12
4.13 TPlayerIsPlaying . . . . .	12
4.14 TPlayerGetCurrentPosition . . . . .	12

4.15 TPlayerGetDuration . . . . .	13
4.16 TPlayerGetMediaInfo . . . . .	13
4.17 TPlayerSetLooping . . . . .	13
4.18 TPlayerSetScaleDownRatio . . . . .	13
4.19 TPlayerSetRotate . . . . .	14
4.20 TPlayerSetSpeed . . . . .	14
4.21 TPlayerSetVolume . . . . .	14
4.22 TPlayerGetVolume . . . . .	15
4.23 TPlayerSetAudioMute . . . . .	15
4.24 TPlayerSetExternalSubUrl . . . . .	15
4.25 TPlayerGetSubDelay . . . . .	15
4.26 TPlayerSetSubDelay . . . . .	16
4.27 TPlayerGetSubCharset . . . . .	16
4.28 TPlayerSetSubCharset . . . . .	16
4.29 TPlayerSwitchAudio . . . . .	16
4.30 TPlayerSwitchSubtitle . . . . .	17
4.31 TPlayerSetSubtitleDisplay . . . . .	17
4.32 TPlayerSetVideoDisplay . . . . .	17
4.33 TPlayerSetDisplayRect . . . . .	17
4.34 TPlayerSetSrcRect . . . . .	18
4.35 TPlayerSetBrightness . . . . .	18
4.36 TPlayerSetContrast . . . . .	18
4.37 TPlayerSetHue . . . . .	19
4.38 TPlayerSetSaturation . . . . .	19
4.39 TPlayerSetEnhanceDefault . . . . .	19
4.40 TPlayerGetVideoDispFramerate . . . . .	19
4.41 TPlayerSetHoldLastPicture . . . . .	20
<b>5 播放器开发流程简单介绍</b>	<b>21</b>
<b>6 播放器开发注意事项</b>	<b>22</b>
<b>7 播放器 tplayerdemo 功能测试</b>	<b>23</b>
7.1 tplayerdemo 测试用例常用的命令说明 . . . . .	23
7.2 tplayerdemo 测试用例剩余所有的命令说明 . . . . .	25
<b>8 播放器 tplayerdemo 压力测试</b>	<b>31</b>

## 插 图

2-1 veConfig . . . . .	2
2-2 ion54Config . . . . .	3
2-3 tplayerConfig . . . . .	4
2-4 tplayerdemoConfig . . . . .	5
3-1 tplayerStatus . . . . .	6
7-1 adbShell . . . . .	23
7-2 tplayerdemo . . . . .	24
7-3 playUrl . . . . .	24
7-4 playUrl . . . . .	25
7-5 setUrl . . . . .	25
7-6 prepare . . . . .	26
7-7 play . . . . .	26
7-8 pause . . . . .	26
7-9 stop . . . . .	27
7-10 seekTo . . . . .	27
7-11 reset . . . . .	27
7-12 quit . . . . .	28
7-13 showMediaInfo . . . . .	28
7-14 showDuration . . . . .	28
7-15 showPosition . . . . .	29
7-16 setLoop . . . . .	29
7-17 setScaleDown . . . . .	29
7-18 fastForward . . . . .	29
7-19 fastBackward . . . . .	30
7-20 getVolume . . . . .	30
7-21 setVolume . . . . .	30
8-1 adbShell . . . . .	31
8-2 tplayerdemoFolder . . . . .	32

# 1 概述

## 1.1 编写目的

此文档说明 Linux 平台，如何使用 TPlayer 的接口来开发播放器应用程序，方便播放器开发人员快速正确地开发，以及播放器测试人员如何根据该文档对 tplayer 播放器进行验证测试。

## 1.2 适用范围

该平台带 ve 模块，支持音频和视频播放。另外，tplayer 播放器支持本地播放和网络播放。网络播放主要支持的流媒体协议：http、https、hls。

## 1.3 相关人员

Linux 平台，tplayer 播放器开发和测试人员。

## 2 软件环境配置

### 2.1 内核配置

在 longon 的根目录执行./build.sh menuconfig。



说明

注：没有 **ve** 模块的芯片（如：**R328**）平台不需要进行内核配置。

#### 2.1.1 选中 **ve** 模块

```
Device Drivers
--> Multimedia support
    -> sunxi video encoder and decoder support
```

如下图所示：

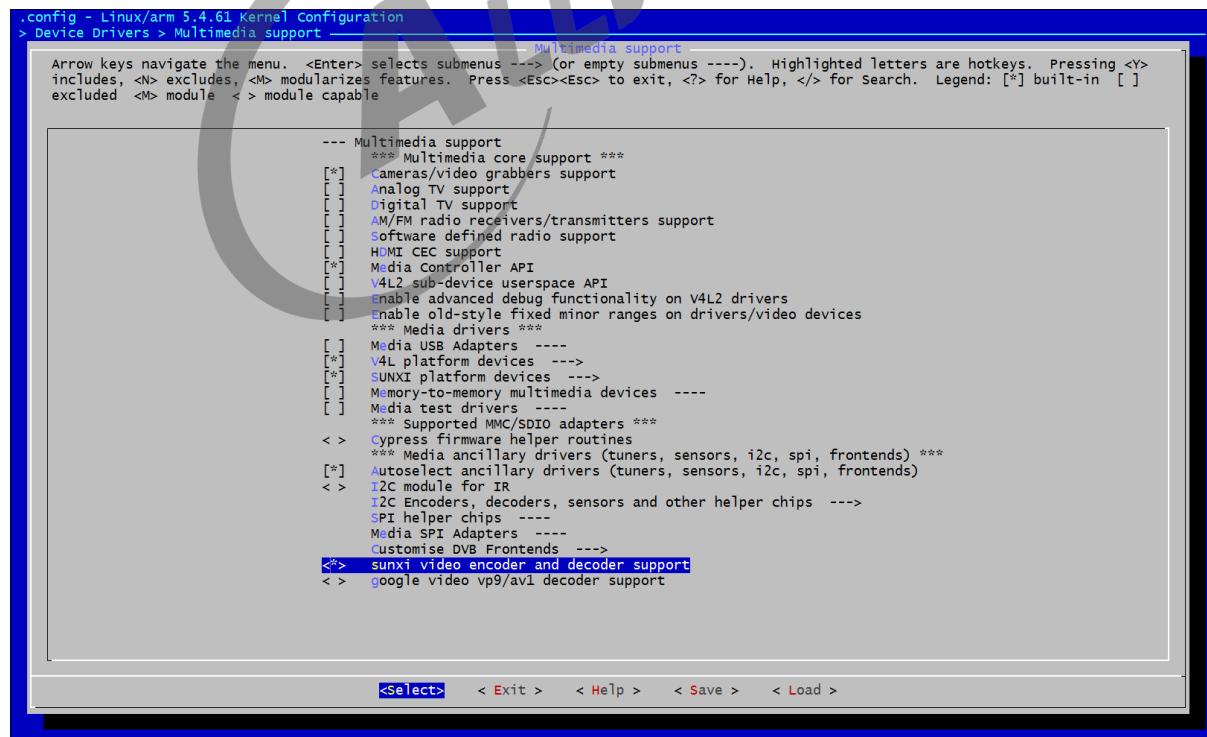


图 2-1: veConfig

## 2.1.2 选中 ion 模块

linux5.4 内核：

```
Device Drivers-->Staging drivers-->Android-->Ion Memory Manager-->Ion system heap  
-->Ion CMA heap support
```

如下图所示：

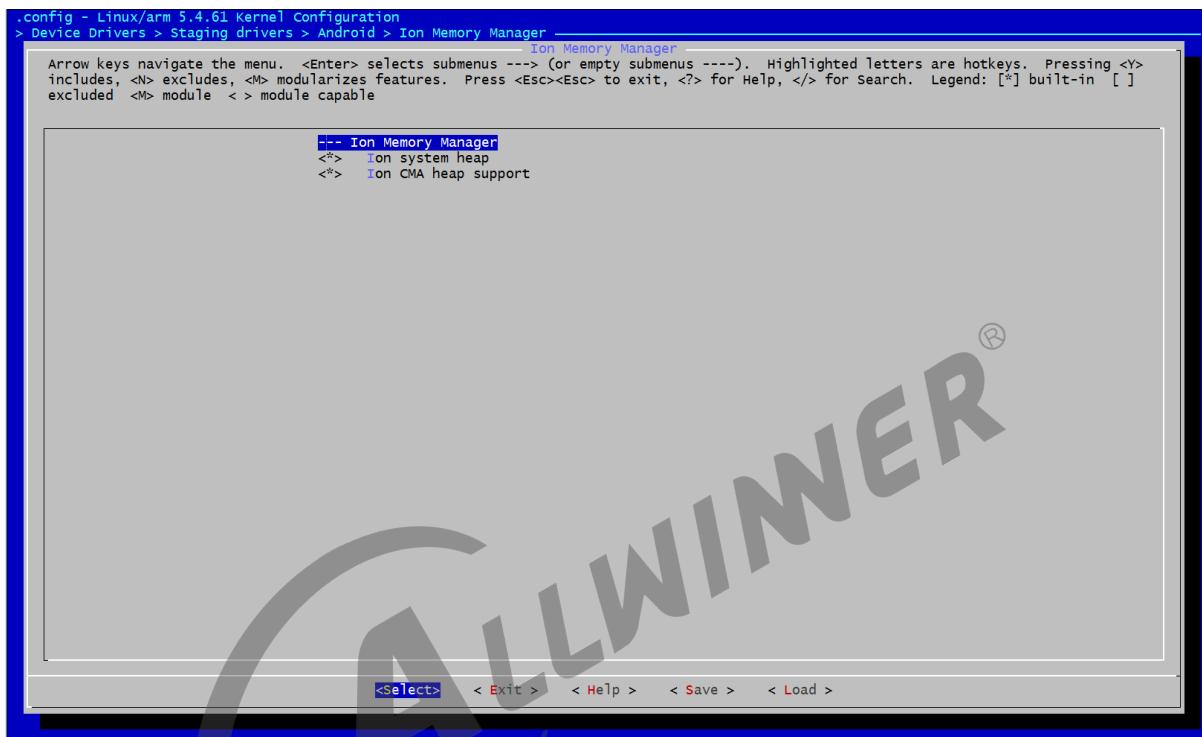


图 2-2: ion54Config

## 2.2 tina 配置

在 tina 的根目录执行 make menuconfig 命令。

### 2.2.1 选中 tplayer 播放中间件

libcedarx：

```
Allwinner-->libcedarx-->Select cedarx configuration options-->Select tplayer middleware
```

如下图所示：

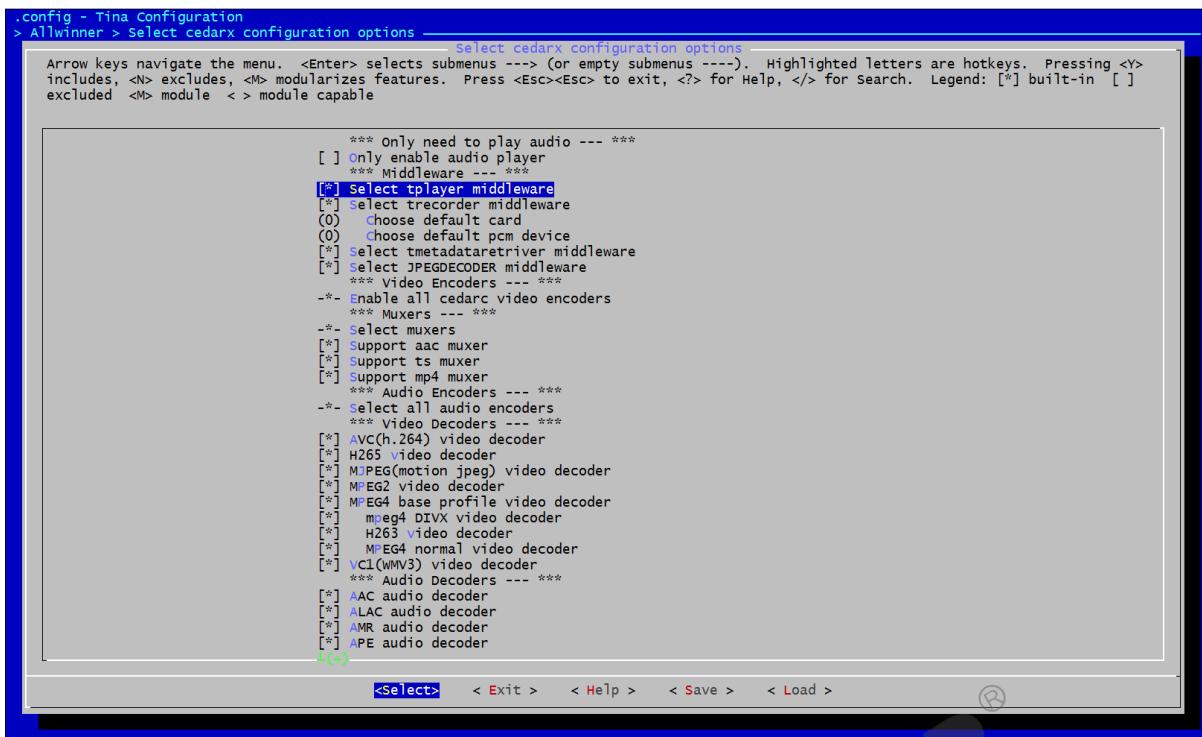


图 2-3: tplayerConfig

## 2.2.2 选中 tplayerdemo

Allwinner-->tina\_multimedia\_demo-->tplayerdemo

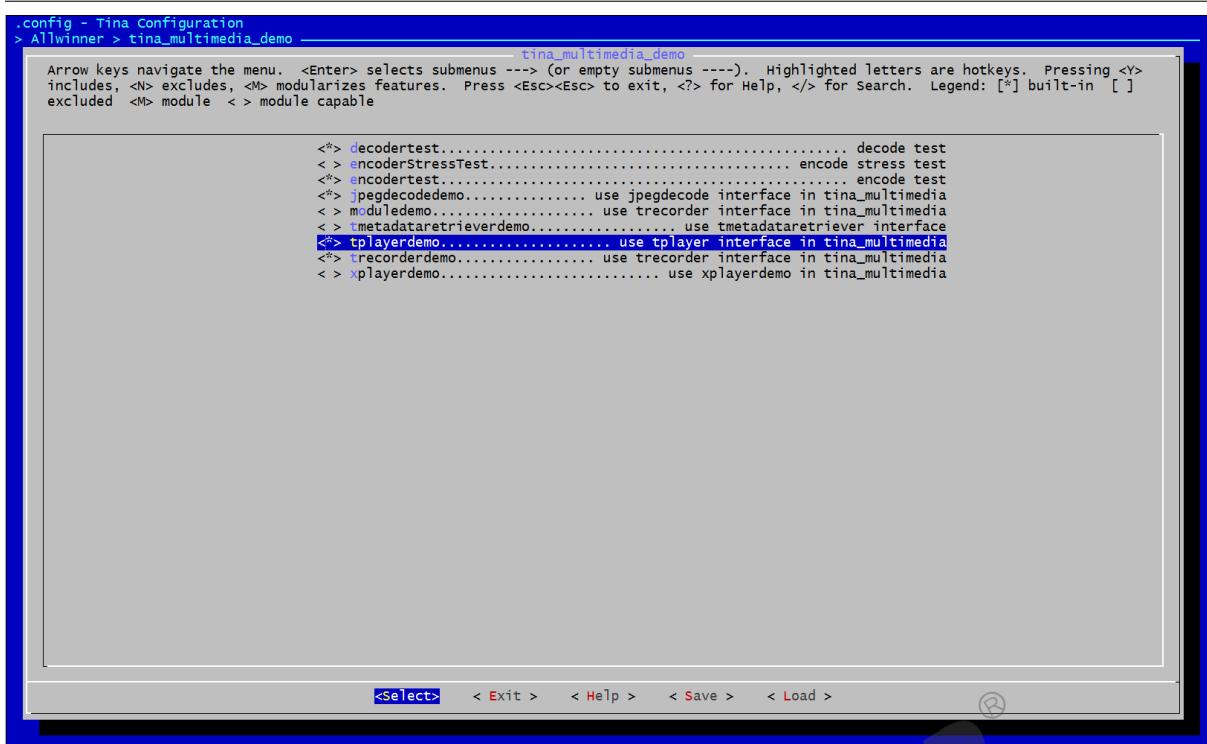


图 2-4: tplayerdemoConfig

### 3 TPlayer 状态图及状态说明

#### 3.1 TPlayer 状态图

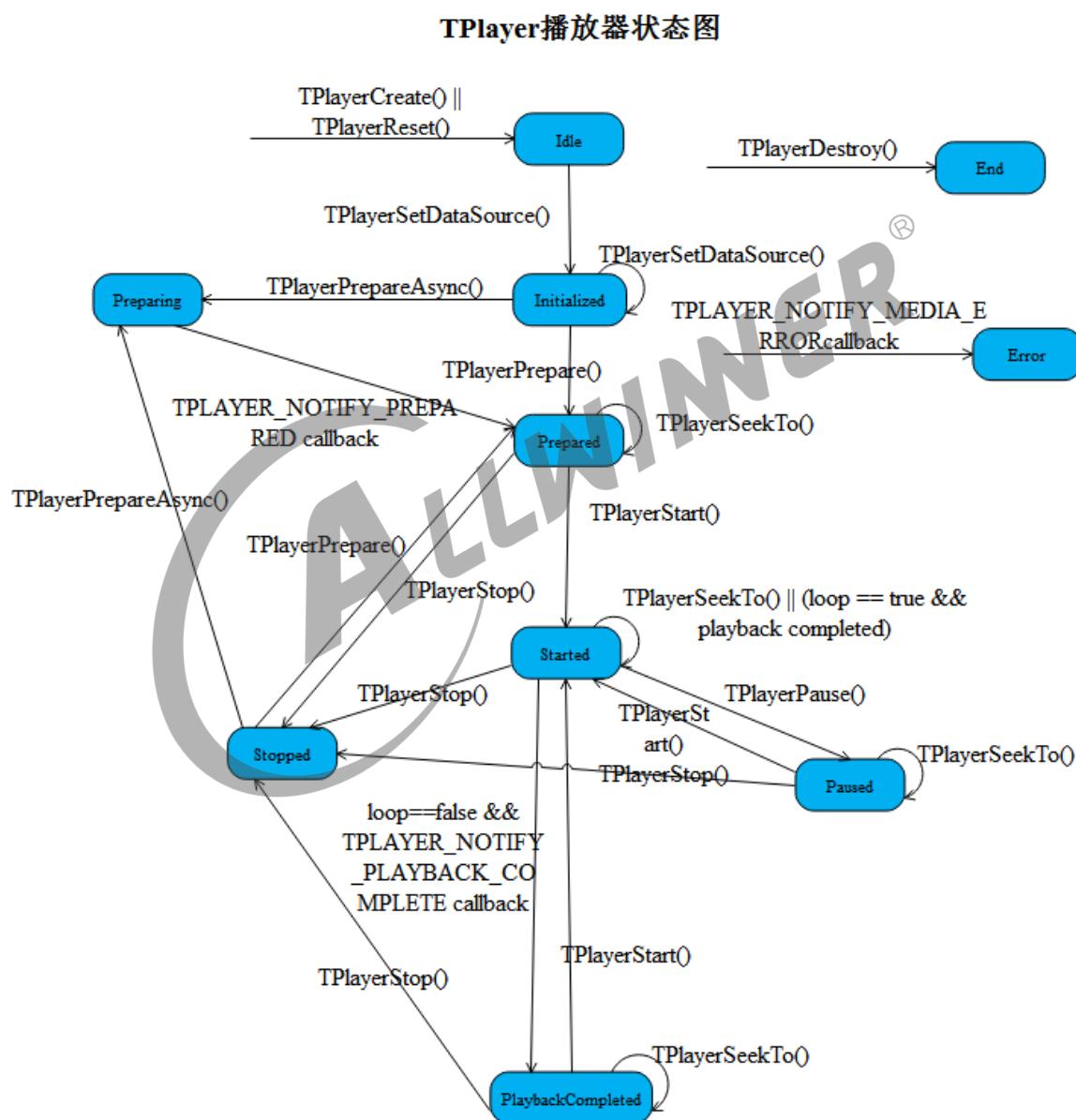


图 3-1: tplayerStatus

这张状态转换图清晰地描述了 TPlayer 的各个状态，也列举了主要的方法的调用时序，每种方法只能在一些特定的状态下使用，否则会出错。另外，只有在 Prepared、Started、Paused、

PlaybackCompleted 这四种状态下可以进行 TPlayerSeekTo() 操作，并且 TPlayerSeekTo() 之后，状态不变。

## 3.2 TPlayer 每个状态简要说明

### 3.2.1 Idle 状态

Idle 状态：当调用 TPlayerCreate() 创建一个 TPlayer 或者调用了其 TPlayerReset() 方法时，TPlayer 处于 idle 状态。

### 3.2.2 Initialized 状态

这个状态比较简单，调用 TPlayerSetDataSource() 方法就进入 Initialized 状态，表示此时要播放的文件已经设置好了。

### 3.2.3 Preparing 状态

调用 TPlayerPrepare() 函数还没返回或者是调用 TPlayerPrepareAsync() 并且还没收到 TPLAYER\_NOTIFY\_PREPARED 这个回调消息的时候就处于 Preparing 状态。

### 3.2.4 Prepared 状态

调用 TPlayerPrepare() 函数已经返回或者是调用 TPlayerPrepareAsync() 并且已经收到 TPLAYER\_NOTIFY\_PREPARED 这个回调消息之后的状态就处于 Prepared 状态。在这个状态下说明所有的资源都已经就绪了，调用 TPlayerStart() 函数就可以播放了。

### 3.2.5 Started 状态

TPlayer 一旦 prepare 完成，就可以调用 TPlayerStart() 方法，这样 TPlayer 就处于 Started 状态，这表明 TPlayer 正在播放文件过程中。可以使用 TPlayerIsPlaying() 测试 TPlayer 是否处于了 Started 状态。如果播放完毕，而又设置了循环播放，则 TPlayer 仍然会处于 Started 状态。

### 3.2.6 Paused 状态

Started 状态下可以调用 TPlayerPause() 方法暂停 TPlayer，从而进入 Paused 状态，TPlayer 暂停后再次调用 TPlayerStart() 则可以继续 TPlayer 的播放，转到 Started 状态。

### 3.2.7 Stopped 状态

Started 或者 Paused 状态下均可调用 TPlayerStop() 停止 TPlayer，而处于 Stop 状态的 TPlayer 要想重新播放，需要通过 TPlayerPrepareAsync() 和 TPlayerPrepare() 回到先前的 Prepared 状态重新开始才可以。

### 3.2.8 PlaybackCompleted 状态

文件正常播放完毕，而又没有设置循环播放的话就进入该状态，并且会通过 TPLAYER\_NOTIFY\_PLAYBACK\_COMPLETE 这个消息回调给应用。此时可以调用 TPlayerStart() 方法重新从头播放文件，也可以 TPlayerStop() 停止 TPlayer，或者也可以 TPlayerSeekTo() 来重新定位播放位置。

### 3.2.9 Error 状态

由于某种原因 TPlayer 出现了错误，就会进入该状态，并且会通过 TPLAYER\_NOTIFY\_MEDIA\_ERROR 这个消息回调给应用。如果 TPlayer 进入了 Error 状态，可以通过调用 TPlayerReset() 来恢复，使得 TPlayer 重新返回到 Idle 状态。

### 3.2.10 End 状态

通过 TPlayerDestroy() 的方法可以进入 End 状态，只要 TPlayer 不再被使用，就应当尽快将其 destroy 掉。

## 4 接口函数说明

### 4.1 TPlayerCreate

函数原型	TPlayer* TPlayerCreate(TplayerType type);
功能	创建一个 TPlayer
参数	type: 底层实际对接的播放器类型，有 CEDARX_PLAYER 和 AUDIO_PLAYER 这两种，AUDIO_PLAYER 只能播放音频
返回值	成功返回 TPlayer 的指针，失败返回 NULL
调用说明	无
备注	type 如果是传入 CEDARX_PLAYER，但是也只想播放音频，那么可以在 make menuconfig 的时候选上 Allwinner->Select cedarx configuration options —> Only enable audio player

### 4.2 TPlayerDestroy

函数原型	void TPlayerDestroy(TPlayer* p);
功能	销毁一个 TPlayer
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针
返回值	无
调用说明	无

### 4.3 TPlayerSetDebugFlag

函数原型	int TPlayerSetDebugFlag(TPlayer* p,bool debugFlag);
功能	设置是否打开调试信息
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; debugFlag: 打开调试信息 标志,true 为打开,false 为关闭
返回值	成功返回 0，失败返回-1
调用说明	目前该函数还没有实现

## 4.4 TPlayerSetNotifyCallback

---

函数原型	int TPlayerSetNotifyCallback(TPlayer* p, TPlayerNotifyCallback notifier, void* pUserData);
功能	设置 TPlayer 的消息回调函数
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; notifier: 回调消息处理函数指针, 需要由应用实现; pUserData: 回调消息处理对象
返回值	成功返回 0, 失败返回-1
调用说明	创建完 TPlayer 播放器之后, 就要调用该函数设置回调消息处理函数。

---

## 4.5 TPlayerSetDataSource

---

函数原型	int TPlayerSetDataSource(TPlayer* p, const char* pUrl, const CdxKeyedVectorT* pHolders);
功能	设置播放文件的 url, 可以是本地文件也可以是网络源
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; pUrl: 需要播放的文件的 url; pHolders: http 的一些头部信息
返回值	成功返回 0, 失败返回-1
调用说明	无

---

## 4.6 TPlayerPrepare

---

函数原型	int TPlayerPrepare(TPlayer* p);
功能	解析文件头部信息, 获取元数据
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针
返回值	成功返回 0, 失败返回-1
调用说明	该函数是阻塞函数, 调用完返回之后就进入了 Prepared 状态, 此时可调 TPlayerStart() 函数进行播放

---

## 4.7 TPlayerPrepareAsync

---

函数原型	int TPlayerPrepareAsync(TinaPlayer* p);
功能	解析文件头部信息, 获取元数据
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针

---

函数原型 int TPlayerPrepareAsync(TinaPlayer\* p);

返回值 成功返回 0，失败返回-1

调用说明 该函数是非阻塞函数，需要等到 TPLAYER\_NOTIFY\_PREPARED 消息回调之后才能调 TPlayerStart() 函数进行播放，而且 TPlayerStart() 函数不能在回调函数中调用

---

## 4.8 TPlayerStart

---

函数原型 int TinaPlayerStart(TPlayer\* p);

功能 开始播放

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针

返回值 成功返回 0，失败返回-1

调用说明 无

---

## 4.9 TPlayerPause

---

函数原型 int TPlayerPause(TPlayer\* p);

功能 暂停播放

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针

返回值 成功返回 0，失败返回-1

调用说明 无

---

## 4.10 TPlayerStop

---

函数原型 int TPlayerStop(TPlayer\* p);

功能 停止播放

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针

返回值 成功返回 0，失败返回-1

调用说明 无

---

## 4.11 TPlayerReset

---

函数原型	int TPlayerReset(TPlayer* p);
功能	重置播放器
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针
返回值	成功返回 0, 失败返回-1
调用说明	在任何状态下都可以调用该函数, 每次播放不同的音频之前, 都需要调用该函数重置播放器, 另外, 一般收到 TPLAYER_NOTIFY_MEDIA_ERROR 这个消息的时候, 也需要通过调用该函数来重置播放器。但是不能在回调函数中调用该函数, 否则会出现死锁

---

## 4.12 TPlayerSeekTo

---

函数原型	int TPlayerSeekTo(TPlayer* p, int nSeekTimeMs);
功能	跳播
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; nSeekTimeMs: 跳播的位置, 单位是 ms
返回值	成功返回 0, 失败返回-1
调用说明	无

---

## 4.13 TPlayerIsPlaying

---

函数原型	bool TPlayerIsPlaying(TPlayer* p);
功能	是否正在播放
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针
返回值	返回 true 表示正在播放, 返回 false 表示没在播放
调用说明	无

---

## 4.14 TPlayerGetCurrentPosition

---

函数原型	int TPlayerGetCurrentPosition(TPlayer* p, int* msec);
功能	获取当前播放的位置
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; msec: 存放当前播放的位置值, 单位: ms
返回值	成功返回 0, 失败返回-1
调用说明	无

---

## 4.15 TPlayerGetDuration

函数原型	int TPlayerGetDuration(TPlayer* p, int* msec);
功能	获取播放的文件总时长
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; msec: 存储文件总时长, 单位: ms
返回值	成功返回 0, 失败返回-1
调用说明	需要在 prepared 状态之后才可以调用该函数

## 4.16 TPlayerGetMediaInfo

函数原型	int TPlayerGetMediaInfo(TPlayer* p, MediaInfo* mediaInfo);
功能	获取媒体信息
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; mediaInfo: 存储媒体信息的指针
返回值	成功返回 0, 失败返回-1。如果失败, 则 mediaInfo 指针为 NULL
调用说明	需要在 prepared 状态之后才可以调用该函数

## 4.17 TPlayerSetLooping

函数原型	int TPlayerSetLooping(TPlayer* p, bool bLoop);
功能	设置循环播放模式
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; bLoop: true 表示单曲循环, false 表示不会单曲循环
返回值	成功返回 0, 失败返回-1。
调用说明	无

## 4.18 TPlayerSetScaleDownRatio

函数原型	int TPlayerSetScaleDownRatio(TPlayer* p, TplayerVideoScaleDownType nHorizonScaleDown, TplayerVideoScaleDownType nVerticalScaleDown);
功能	设置视频的水平方向的缩放比例和垂直方向的缩放比例

---

函数原型	int TPlayersetScaleDownRatio(TPlayer* p,TplayerVideoScaleDownType nHorizonScaleDown, TplayerVideoScaleDownType nVerticalScaleDown);
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; nHorizonRatio: 水平方向的缩放比例; nVerticalRatio: 垂直方向的缩放比例
返回值	成功返回 0, 失败返回-1
调用说明	这个函数需要在 prepare 之前调用

---

## 4.19 TPlayerSetRotate

---

函数原型	int TPlayerSetRotate(TPlayer* p,TplayerVideoRotateType rotateDegree);
功能	设置视频旋转的角度
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; rotateDegree: 视频旋转的角度
返回值	成功返回 0, 失败返回-1。
调用说明	这个函数需要在 TPlayerSetDataSource() 函数之前调用

---

## 4.20 TPlayerSetSpeed

---

函数原型	int TPlayerSetSpeed(TPlayer* p, TplayerPlaySpeedType nSpeed);
功能	设置快进快退的速度
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; nSpeed: 快进快退的速度
返回值	成功返回 0, 失败返回-1。
调用说明	无

---

## 4.21 TPlayerSetVolume

---

函数原型	int TPlayerSetVolume(TPlayer* p,int volume);
功能	设置播放器的音量大小
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; volume: 需要设置的音量大小值
返回值	成功返回 0, 失败返回-1。
调用说明	无

---

## 4.22 TPlayerGetVolume

函数原型	int TPlayerGetVolume(TPlayer* p,int* volume);
功能	获取播放器当前音量的大小
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; volume: 存放获取的音量的大小值
返回值	成功返回 0, 失败返回-1。
调用说明	无

## 4.23 TPlayerSetAudioMute

函数原型	int TPlayerSetAudioMute(TPlayer* p,bool mute);
功能	静音或不静音
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; mute: 是否静音, true 为静音, false 为不静音
返回值	成功返回 0, 失败返回-1
调用说明	无

## 4.24 TPlayerSetExternalSubUrl

函数原型	int TPlayerSetExternalSubUrl(TPlayer* p, const char* filePath);
功能	设置外挂字幕的路径
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; filePath: 外挂字幕的路径
返回值	成功返回 0, 失败返回-1。
调用说明	无

## 4.25 TPlayerGetSubDelay

函数原型	int TPlayerGetSubDelay(TPlayer* p);
功能	获取字幕提前或延时的时间
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针
返回值	成功返回字幕提前或延时的时间, 单位: ms, 失败返回-1
调用说明	无

## 4.26 TPlayerSetSubDelay

函数原型	int TPlayerSetSubDelay(TPlayer* p, int nTimeMs);
功能	设置字幕提前或延时的时间
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; nTimeMs: 字幕延时或提前的时间
返回值	成功返回 0, 失败返回-1。
调用说明	无

## 4.27 TPlayerGetSubCharset

函数原型	int TPlayerGetSubCharset(TPlayer* p, char *charset);
功能	获取字幕的字符编码格式
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; charset: 获取字幕的编码格式
返回值	成功返回 0, 失败返回-1。
调用说明	无

## 4.28 TPlayerSetSubCharset

函数原型	int TPlayerSetSubCharset(TPlayer* p, const char* strFormat);
功能	设置字幕的字符编码格式
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; strFormat: 字幕的编码格式
返回值	成功返回 0, 失败返回-1。
调用说明	无

## 4.29 TPlayerSwitchAudio

函数原型	int TPlayerSwitchAudio(TPlayer* p, int nStreamIndex);
功能	切换音轨
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; nStreamIndex: 音轨的 index
返回值	成功返回 0, 失败返回-1。

---

函数原型 int TPlayerSwitchAudio(TPlayer\* p, int nStreamIndex);

---

调用说明 无

---

## 4.30 TPlayerSwitchSubtitle

---

函数原型 int TPlayerSwitchSubtitle(TPlayer\* p, int nStreamIndex);

---

功能 切换字幕

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; nStreamIndex: 字幕流的 index

返回值 成功返回 0, 失败返回-1。

调用说明 无

---

## 4.31 TPlayerSetSubtitleDisplay

---

函数原型 void TPlayerSetSubtitleDisplay(TPlayer\* p, bool onoff);

---

功能 设置打开或关闭字幕

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; onoff: 打开或关闭字幕的标志位

返回值 无

调用说明 无

---

## 4.32 TPlayerSetVideoDisplay

---

函数原型 void TPlayerSetVideoDisplay(TPlayer\* p, bool onoff);

---

功能 设置是否显示视

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; onoff: 是否显示视频的标志位

返回值 无

调用说明 无

---

## 4.33 TPlayerSetDisplayRect

---

函数原型	void TPlayerSetDisplayRect(TPlayer* p,int x, int y, unsigned int width, unsigned int height);
功能	设置显示的区域
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; x: 显示区域起始点 x 坐标的值; y: 显示区域起始点 y 坐标的值; width: 显示区域的宽; height: 显示区域的高
返回值	无
调用说明	无

---

## 4.34 TPlayerSetSrcRect

---

函数原型	void TPlayerSetSrcRect(TPlayer* p,int x, int y, unsigned int width, unsigned int height);
功能	设置原图像 crop 的坐标和大小
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; x: 源图像 crop 起始点 x 坐标的值; y: 源图像 crop 起始点 y 坐标的值; width: 源图像 crop 的宽; height: 源图像 crop 的高
返回值	无
调用说明	无

---

## 4.35 TPlayerSetBrightness

---

函数原型	void TPlayerSetBrightness(TPlayer* p,unsigned int grade);
功能	设置亮度
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; grade: 亮度的值, 范围是:0-100, 默认值是:50
返回值	无
调用说明	无

---

## 4.36 TPlayerSetContrast

---

函数原型	void TPlayerSetContrast(TPlayer* p,unsigned int grade);
功能	设置对比度
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; grade: 对比度的值, 范围是:0-100, 默认值是:50

---

---

函数原型	void TPlayerSetContrast(TPlayer* p,unsigned int grade);
返回值	无
调用说明	无, 注：这个接口没实现

---

## 4.37 TPlayerSetHue

---

函数原型	void TPlayerSetHue(TPlayer* p,unsigned int grade);
功能	设置色调
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; grade: 色调的值，范围是:0-100，默认值是:50
返回值	无
调用说明	无, 注：这个接口没实现

---

## 4.38 TPlayerSetSaturation

---

函数原型	void TPlayerSetSaturation(TPlayer* p,unsigned int grade);
功能	设置饱和度
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; grade: 饱和度的值，范围是:0-100，默认值是:50
返回值	无
调用说明	无, 注：这个接口没实现

---

## 4.39 TPlayerSetEnhanceDefault

---

函数原型	void TPlayerSetEnhanceDefault(TPlayer* p);
功能	使能丽色系统，并用默认设置的值
参数	p: 通过 TPlayerCreate 函数创建的 TPlayer 指针
返回值	无
调用说明	无

---

## 4.40 TPlayerGetVideoDispFramerate

---

函数原型 int TPlayerGetVideoDispFramerate(TPlayer\* p, float\* dispFramerate);

功能 获取显示的帧率

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; dispFramerate: 目前播放实际帧率的值存放在哪里

返回值 成功返回 0, 失败返回-1。

调用说明 该函数只能在播放过程中调用

---

## 4.41 TPlayerSetHoldLastPicture

---

函数原型 int TPlayerSetHoldLastPicture(TPlayer\* p, int bHoldFlag);

功能 播放完之后是否保留最后一帧画面在屏幕上

参数 p: 通过 TPlayerCreate 函数创建的 TPlayer 指针; bHoldFlag: 1 表示需要保留最后一帧, 0 表示不需要保留最后一帧

返回值 成功返回 0, 失败返回-1。

调用说明 该函数需要在 prepare 之后, start 之前调用

---

## 5 播放器开发流程简单介绍

- (1) TPlayerCreate() //创建一个播放器
- (2) TPlayerSetNotifyCallback() //设置消息回调函数
- (3) TPlayerSetDataSource() //设置 url
- (4) TPlayerPrepare() 或 TPlayerPrepareAsync() //解析头部信息，获取元数据，并根据元数据的信息初始化对应的解码器
- (5) TPlayerStart() //播放 (注：如果是用 TPlayerPrepareAsync() 函数，则需要等到 TPLAYER\_NOTIFY\_PREPARED 消息回调之后才可以调用 TPlayerStart() 函数进行播放)
- (6) 如果需要跳播，则可以调用 TPlayerSeekTo() 函数
- (7) 如果需要暂停，则调用 TPlayerPause() 函数进行暂停
- (8) 如果需要停止，则可以调用 TPlayerStop() 或 TPlayerReset() 函数进行停止 (注：建议用 TPlayerReset() 函数进行停止，因为任何状态下都可以调用 TPlayerReset() 函数)
- (9) 如果需要播放下一个或其他的，则可以先调用 TPlayerReset() 函数使播放器进入 idle 状态，然后再重复 (3)(4)(5) 的步骤
- (10) 详细的播放器开发 demo 可以参考这个路径的内容：  
package/allwinner/tina\_multimedia\_demo/tplayerdemo

## 6 播放器开发注意事项

(1) TPlayerSetNotifyCallback(TPlayer\* p, TPlayerNotifyCallback notifier, void\* pUserData) 函数必需要调用，而且 notifier 不能为 NULL。

(2) 回调函数中不能调用 TPlayer 的任何一个接口，如：TPlayerReset、TPlayerStop、TPlayerStart 等这些接口不能在回调函数中调用。



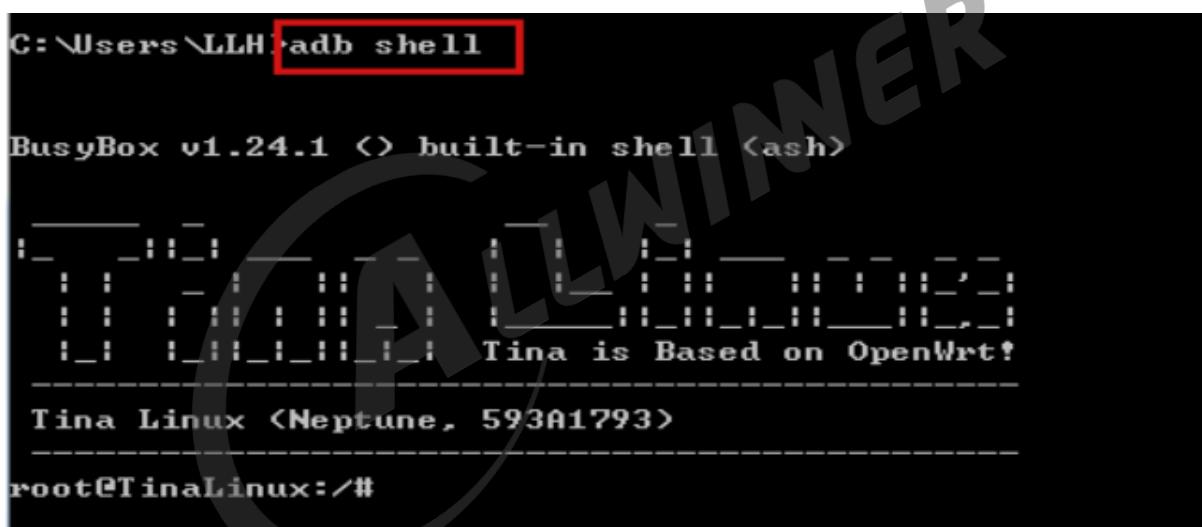
## 7 播放器 tplayerdemo 功能测试

### 7.1 tplayerdemo 测试用例常用的命令说明

编译固件之前，如果已经把 tplayerdemo 测试用例选上的话，编译出来的固件就可以使用 tplayerdemo 来测试音视频播放的基本功能了。具体的操作如下：

(1) 把开发板用 usb 线和电脑连接起来，并且确保 adb 能用。如果 adb 不能用，也可以用串口线来代替。

(2) 执行 adb shell 命令：



The screenshot shows a terminal window with the following text:

```
C:\Users\LLH>adb shell
BusyBox v1.24.1 <-- built-in shell (ash)
Tina is Based on OpenWrt!
Tina Linux (Neptune, 593A1793)
root@TinaLinux:/#
```

图 7-1: adbShell

(3) 运行 tplayerdemo 测试程序：

```
root@TinaLinux:/# tplayerdemo
tplayerdemo
set MR100 audio throughway
numid=17,iface=MIXER,name='Speaker Function'
: type=ENUMERATED,access=rw----,values=1,items=3
: Item #0 'headset'
: Item #1 'spk'
: Item #2 'headset-spk'
: values=2
numid=1,iface=MIXER,name='Master Playback Volume'
: type=INTEGER,access=rw----,values=1,min=0,max=63,step=0
: values=40
WARNING: awplayer <log_set_level:30>: Set log level to 3
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-0 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-1 ok.
```

图 7-2: tplayerdemo

(4) 播放某个本地或网络音视频：play url: 本地文件的绝对路径或网络 url, 例如需要播放的视频为 h264.mp4，并且该视频放在/mnt/UDISK/目录下，则命令为：

```
tplayerdemo# play url:/mnt/UDISK/h264.mp4
play url:/mnt/UDISK/h264.mp4

tplayerdemo# demoPlayer.mUrl = /mnt/UDISK/h264.mp4
WARNING: awplayer <XPlayerReset:946>: reset...
DEBUG : awplayer <PlayerStop:850>: ***** PlayerStop
ERROR : awplayer <PlayerStop:855>: +[40;31minvalid stop operation, player already in stopped status.+[0m
reset the player ok.
DEBUG : awplayer <XPlayerSetDataSourceUrl:457>: setDataSource(url), url='/mnt/UDISK/h264.mp4'
INFO : awplayer <XPlayerThread:1731>: process message XPLAYER_COMMAND_SET_SOURCE.
DEBUG : awplayer <XPlayerPrepare:742>: prepare
DEBUG : awplayer <XPlayerThread:1984>: process message XPLAYER_COMMAND_PREPARE. mPriData->mStatus: 1
DEBUG : demuxComponent <DenuxThread:1784>: process message DEMUX_COMMAND_PREPARE.
DEBUG : demuxComponent <DenuxThread:1851>: == prepare msg
DEBUG : awplayer <CdxParserPrepare:728>: source uri 'file:///mnt/UDISK/h264.mp4'
DEBUG : awplayer <__FileStreamCreate:533>: local file 'file:///mnt/UDISK/h264.mp4'
```

图 7-3: playUrl

(5) 注：有一条命令可以直接播放一个本地文件，例如需要播放的视频为 h264\_aac\_30fps.mp4，并且该视频放在/mnt/UDISK/目录下，则命令为：

```
root@TinaLinux:/# tplayerdemo /mnt/UDISK/h264_aac_30fps.mp4
tplayerdemo /mnt/UDISK/h264_aac_30fps.mp4
set r16 audio pass through
numid=1,iface=MIXER,name='headphone volume'
: type=INTEGER,access=rw---R--,values=1,min=0,max=63,step=0
: values=30
: dBscale-min=-63.00dB,step=1.00dB,mute=0
numid=100,iface=MIXER,name='DACL Mixer AIF1DA0L Switch'
: type=BOOLEAN,access=rw-----,values=1
: values=on
numid=96,iface=MIXER,name='DACR Mixer AIF1DA0R Switch'
: type=BOOLEAN,access=rw-----,values=1
: values=on
numid=105,iface=MIXER,name='Headphone Switch'
: type=BOOLEAN,access=rw-----,values=1
: values=on
WARNING: awplayer <log_set_level:30>: Set log level to 3
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-0 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-1 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-2 ok.
```

图 7-4: playUrl

## 7.2 tplayerdemo 测试用例剩余所有的命令说明

(1) 执行 set url: 命令设置 url, 其中 set url: 是固定的格式, 冒号后面跟本地文件的绝对路径或网络 url 路径:

```
tplayerdemo# set url:/mnt/SDCARD/avc_aac.flv
set url:/mnt/SDCARD/avc_aac.flv

tplayerdemo# demoPlayer.mUrl = /mnt/SDCARD/avc_aac.flv
DEBUG : amplayer <XPlayerSetDataSourceUrl:426>: setDataSource(url), url='/mnt/SDCARD/avc_aac.flv'
INFO : amplayer <XPlayerThread:1575>: process message XPLAYER_COMMAND_SET_SOURCE.
DEBUG : amplayer <XPlayerPrepare:686>: prepare
DEBUG : amplayer <XPlayerThread:1792>: process message XPLAYER_COMMAND_PREPARE. nPriData->nStatus: 1
DEBUG : demuxComponent <DemuxThread:1693>: process message DEMUX_COMMAND_PREPARE.
DEBUG : demuxComponent <DemuxThread:1768>: === prepare msg
DEBUG : amplayer <CdxParserPrepare:711>: source uri 'file:///mnt/SDCARD/avc_aac.flv'
DEBUG : amplayer <_FileStreamCreate:528>: local file 'file:///mnt/SDCARD/avc_aac.flv'
DEBUG : amplayer <_FileStreamConnect:387>: *****impl->size=19835335
DEBUG : amplayer <_FileStreamConnect:399>: impl->filePath=fd://$offset=0&length=19835335
DEBUG : amplayer <_FileStreamConnect:481>: :16:[00 00 00 18 66 74 79 78 69 73 6f 6d 00 00 00 01]
```

图 7-5: setUrl

(2) 执行 prepare 命令解析需要播放的数据:

```
tplayerdemo# prepare
prepare

tplayerdemo# DEBUG : awplayer <XPlayerPrepareSync:666>: prepareSync
DEBUG : awplayer <XPlayerThread:1792>: process message XPLAYER_COMMAND_PREPARE. mPriData->mStatus: 4
INFO : awplayer <XPlayerThread:1828>: xxxxxxxxxxxx video size: width = 640, height = 480
DEBUG : tplayer <CallbackFromXPlayer:82>: video width = 640,height = 480
warning: unknown callback from Tinaplayer.
TPLAYER_NOTIFY_PREPARED, has prepared.
prepare
prepared ok
```

图 7-6: prepare

(3) 执行 play 命令播放。注：如果是播放本地文件，则可以省略 prepare 这个步骤，直接 set url 完成之后，执行 play 命令进行播放。

```
tplayerdemo# play
play

tplayerdemo# DEBUG : awplayer <XPlayerStart:716>: start
DEBUG : awplayer <XPlayerThread:1972>: process message XPLAYER_COMMAND_START.
DEBUG : awplayer <PlayerStart:731>: player start
DEBUG : awplayer <BaseCompPostAndWait:61>: video decoder receive cmd: start
DEBUG : awplayer <BaseCompPostAndWait:61>: audio decoder receive cmd: start
error : AllwinnerAudioCodec <CreateAudioDecoder:1170>: +[40;31mCreate Decoder!!====+[0m
error : AllwinnerAudioCodec <InitCodec:280>: +[40;31mKhan----Loading 'libaw_aacdec.so' success![0m
DEBUG : awplayer <BaseCompPostAndWait:61>: video render receive cmd: start
DEBUG : awplayer <BaseCompPostAndWait:61>: audio render receive cmd: start
INFO : audioRender <HandleStart:295>: audio render process start message.
DEBUG : audioRender <InitSoundDevice:478>: init sound device.
```

图 7-7: play

(4) 执行 pause 命令暂停：

```
tplayerdemo# DEBUG : awplayer <QueueBufferToShow:1247>: video pts(35.002)
pause
pause

tplayerdemo# DEBUG : awplayer <XPlayerPause:778>: pause
DEBUG : awplayer <BaseCompPostAndWait:61>: video render receive cmd: pause
DEBUG : awplayer <BaseCompPostAndWait:61>: audio render receive cmd: pause
INFO : audioRender <HandlePause:363>: audio render process pause message.
DEBUG : audioRender <HandlePause:376>: pause sound devide.
DEBUG : tsoundcontrol <TSoundDevicePause:292>: TinaSoundDevicePause(): sc->sound_status = 0
DEBUG : tsoundcontrol <TSoundDevicePause:303>: alsa can not pause,use snd_pcm_drop
DEBUG : awplayer <BaseCompPostAndWait:61>: video decoder receive cmd: pause
DEBUG : awplayer <BaseCompPostAndWait:61>: audio decoder receive cmd: pause
paused.
```

图 7-8: pause

(5) 执行 stop 命令停止：

```
tplayerdemo# stop
stop

tplayerdemo# DEBUG : awplayer <XPlayerStop:758>: stop
DEBUG : CdxMovParser <__CdxMovParserForceStop:943>: -- mov ForceStop end
DEBUG : awplayer <CdxMovClose:5595>: mov close stream = 0x249a0
DEBUG : awplayer <PlayerStop:843>: ***** PlayerStop
DEBUG : awplayer <BaseCompPostAndWait:61>: audio render receive cmd: stop
INFO : audioRender <handleStop:336>: audio render process stop message.
DEBUG : audioRender <handleStop:349>: stop sound devide.
DEBUG : tsoundcontrol <TSoundDeviceStop:260>: TinaSoundDeviceStop():sc->sound_status = 1
DEBUG : tsoundcontrol <closeSoundDevice:42>: closeSoundDevice()
```

图 7-9: stop

(6) 执行 seek to: 命令进行跳播，其中 seek to: 是固定格式，冒号后面跟跳播的时间，单位是秒。如跳播到 100 秒，则用以下这条命令：

```
tplayerdemo# DEBUG : awplayer <QueueBufferToShow:1247>: video pts<7.007>
seedebug : cedar <savePicture:1349>: saving picture, size: 640 x 480, format: 5, count: 3
DEBUG : awplayer <QueueBufferToShow:1247>: video pts<8.008>
k toDEBUG : awplayer <QueueBufferToShow:1247>: video pts<9.009>
:1DEBUG : awplayer <QueueBufferToShow:1247>: video pts<10.010>
00
seek to:100

tplayerdemo# nSeekTimeMs = 100000 , nDuration = 184617
DEBUG : awplayer <XPlayerSeekTo:818>: seekTo [100000ms]
DEBUG : awplayer <XPlayerSeekTo:867>: seek
```

图 7-10: seekTo

(7) 执行 reset 命令重置播放器

```
tplayerdemo# reset
reset

tplayerdemo# WARNING: awplayer <XPlayerReset:887>: reset...
DEBUG : CdxMovParser <__CdxMovParserForceStop:943>: -- mov ForceStop end
DEBUG : awplayer <CdxMovClose:5595>: mov close stream = 0xbcc150
DEBUG : awplayer <PlayerStop:843>: ***** PlayerStop
DEBUG : awplayer <BaseCompPostAndWait:61>: audio render receive cmd: stop
INFO : audioRender <handleStop:336>: audio render process stop message.
DEBUG : audioRender <handleStop:349>: stop sound devide.
DEBUG : tsoundcontrol <TSoundDeviceStop:260>: TinaSoundDeviceStop():sc->sound_status = 1
DEBUG : tsoundcontrol <closeSoundDevice:42>: closeSoundDevice()
ERROR : tsoundcontrol <closeSoundDevice:52>: +[40;31malsa-uninit: pcm closed+[0m
DEBUG : awplayer <BaseCompPostAndWait:61>: video render receive cmd: stop
DEBUG : awplayer <__LayerCtrlHoldLastPicture:368>: LayerCtrlHoldLastPicture, bHold = 0
DEBUG : awplayer <BaseCompPostAndWait:61>: audio decoder receive cmd: stop
debug : AllwinnerAudioCodec <ExitCodec:329>: Khan----dlclose libaw_aacdec.so success!
error : AllwinnerAudioCodec <DestroyAudioDecoder:1090>: +[40;31mdestroy_ResampleInfo fail!!+[0m
DEBUG : awplayer <BaseCompPostAndWait:61>: video decoder receive cmd: stop
DEBUG : awplayer <PlayerStop:875>: ***** PlayerStop end
DEBUG : awplayer <BaseCompPostAndWait:61>: video render receive cmd: quit
```

图 7-11: reset

(8) 执行 quit 命令退出 tplayerdemo 程序或者直接 ctrl+c 退出 tplayerdemo 程序

```
tplayerdemo# quit
quit

tplayerdemo# COMMAND_QUIT
destroy TinaPlayer.
WARNING: awplayer <XPlayerDestroy:297>: XPlayerDestroy
WARNING: awplayer <XPlayerReset:887>: reset...
```

图 7-12: quit

(9) show media info 这条命令可以打印一些媒体信息出来。注：这条命令需要在 prepare 之后运行才有效

```
tplayerdemo# show media info
show media info

tplayerdemo# show media information=
file size = 19370 KB
duration = 184617 ms
bitrate = 839 Kbps
container type = 0
video stream num = 1
audio stream num = 1
subtitle stream num = 0
video codec type = 277
video width = 640
video height = 480
video framerate = 29970
video frameduration = 0
audio codec type = 4
audio channel num = 2
audio BitsPerSample = 16
audio sample rate = 44100
audio bitrate = 0 Kbps
```

图 7-13: showMediaInfo

(10) show duration 这条命令可以打印总时长。注：这条命令需要在 prepare 之后运行才有效

```
tplayerdemo# show duration
show duration

tplayerdemo# media duration = 184 seconds.
```

图 7-14: showDuration

(11) show position 这条命令可以打印当前播放到的时间点。注：这条命令需要在 prepare 之后运行才有效

```
tplayerdemo# show position  
show position  
  
tplayerdemo# current position = 6 seconds.
```

图 7-15: showPosition

(12)set loop 这条命令可以设置是否是单曲循环，如果需要单曲循环，则设为:set loop:1，如果不单曲循环，则设为： set loop:0 . 注：默认不设置的话是为 0 的

```
tplayerdemo# set loop:1  
set loop:1  
  
tplayerdemo# tplayerdemo set loop flag:flag = 1
```

图 7-16: setLoop

(13)set scaledown 这条命令可以设置视频解码后的数据需要缩放的比例，目前支持不缩放、缩放 1/2 和 1/4，jpeg 格式支持缩放到 1/8。如要缩放为原来的 1/2，则用以下命令。注：这个命令需要在 set url 命令前调用

```
tplayerdemo# set scaledown:2  
set scaledown:2  
  
tplayerdemo# tplayerdemo set scaledown value = 2  
scale down 1/2
```

图 7-17: setScaleDown

(14)fast forward 这条命令可以快进。目前支持 2、4、8、16 倍快进，如需要 4 倍快进，则用以下这条命令：

```
tplayerdemo# DEBUG : awplayer <QueueBufferToShow:1247>: video pts(75.008)  
fast forward:4  
fast forward:4  
  
tplayerdemo# tplayerdemo fast forward times = 4  
fast forward 4 times
```

图 7-18: fastForward

(15)fast backward 这条命令可以快退。目前支持 2、4、8、16 倍快退，如需要 4 倍快退，则用以下这条命令：

```
tplayerdemo# DEBUG : awplayer <QueueBufferToShow:1247>: video pts<104.004>
fast backward:4DEBUG : awplayer <QueueBufferToShow:1247>: video pts<105.005>

fast backward:4

tplayerdemo# tplayerdemo fast backward times = 4
fast backward 4 times
```

图 7-19: fastBackward

(16)get volume 这条命令可以获取播放器的音量值。

```
tplayerdemo# get volume
get volume

tplayerdemo# cur volume = 20
```

图 7-20: getVolume

(17)set volume 这条命令可以设置播放器的音量值，支持的音量值范围:0-40，其中 0 表示静音。如：要把播放器音量值设为 30，则可以用下面这条命令：

```
tplayerdemo# set volume:30
set volume:30

tplayerdemo# tplayerdemo setVolume:volume = 30
tplayerdemo set volume ok
```

图 7-21: setVolume

## 8 播放器 tplayerdemo 压力测试

编译固件之前，如果已经把 tplayerdemo 测试用例选上的话，编译出来的固件就可以使用 tplayerdemo 来音视频播放进行压力测试了。具体的操作如下：

- (1) 把开发板用 usb 线和电脑连接起来，并且确保 adb 能用。如果 adb 不能用，也可以用串口线来代替。
- (2) 执行 adb shell 命令：

```
C:\Users\LLH>adb shell
BusyBox v1.24.1 <> built-in shell (ash)
=====
Tina is Based on OpenWrt!
Tina Linux <Neptune, 593A1793>
=====
root@TinaLinux:/#
```

图 8-1: adbShell

- (3) 假设需要循环测试某个目录下的所有音视频文件，则可以用这条命令：tplayerdemo 音视频文件所在的文件夹的绝对路径，假如需要测试的音视频文件放在/mnt/UDISK/目录下，则运行以下命令则可以循环播放/mnt/UDISK/目录下的所有音视频文件：

```
root@TinaLinux:/# tplayerdemo /mnt/UDISK/
tplayerdemo /mnt/UDISK/
set r16 audio pass through
numid=1,iface=MIXER,name='headphone volume'
: type=INTEGER,access=rw---R--,values=1,min=0,max=63,step=0
: values=30
: dBscale--min=-63.00dB,step=1.00dB,mute=0
numid=100,iface=MIXER,name='DACL Mixer AIF1DA0L Switch'
: type=BOOLEAN,access=rw-----,values=1
: values=on
numid=96,iface=MIXER,name='DACR Mixer AIF1DA0R Switch'
: type=BOOLEAN,access=rw-----,values=1
: values=on
numid=105,iface=MIXER,name='Headphone Switch'
: type=BOOLEAN,access=rw-----,values=1
: values=on
WARNING: awplayer <log_set_level:30>: Set log level to 3
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-0 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-1 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-2 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-3 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-4 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-5 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-6 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-7 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-8 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-9 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-10 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-11 ok.
DEBUG : awplayer <ReadPluginEntry:178>: read plugin entry adecoder-12 ok.
```

图 8-2: tplayerdemoFolder

## 著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

## 商标声明

  **全志科技**  (不完全列举) 均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

## 免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。