



Android COMAIR5 Library For Android API Specification

V1.0.5 – Nov. 07, 2014

Important Notice

GENERALPLUS TECHNOLOGY INC. reserves the right to change this documentation without prior notice. Information provided by GENERALPLUS TECHNOLOGY INC. is believed to be accurate and reliable. However, GENERALPLUS TECHNOLOGY INC. makes no warranty for any errors which may appear in this document. Contact GENERALPLUS TECHNOLOGY INC. to obtain the latest version of device specifications before placing your order. No responsibility is assumed by GENERALPLUS TECHNOLOGY INC. for any infringement of patent or other rights of third parties which may result from its use. In addition, GENERALPLUS products are not authorized for use as critical components in life support devices/ systems or aviation devices/systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of Generalplus.

Table of Content

1	Introduction	6
2	Library Module Files	7
2.1	Include Into an Existing Android Project.....	7
2.2	Import the Demo Project.....	8
3	Notice for Developing App	9
3.1	Android Project.....	9
4	Constrains and Limitations	10
5	API Calling Sequence.....	11
5.1	Decode Calling Sequence	11
6	API Specification	12
6.1	Data Structure	12
6.2	APIs Provided by the Wrapper Class	13
6.2.1	StartComAir5DecodeProcess	13
6.2.2	StopComAir5DecodeProcess.....	13
6.2.3	PlayComAirCmd	14
6.2.4	PlayComAirCmdList	14
6.2.5	IsComAir5CmdPlaying	15
6.2.6	CalibrateComAir5Recorder	15
6.2.7	GetComAirRecorderResourceType	15
6.3	Native APIs.....	16
6.3.1	InitComAir5	16
6.3.2	UnitComAir5	16
6.3.3	StartComAir5Decode	16
6.3.4	StopComAir5Decode.....	17
6.3.5	SetComAir5Property	17
6.3.6	GetComAir5Proterty.....	18
6.3.7	ComAir5Decode.....	18
6.3.8	PlayComAir5Cmd	19
6.3.9	PlayComAir5CmdList	19
6.4	COMAIR5 Property Type.....	20
6.4.1	eComAir5Property_RegCode.....	20
6.4.2	eComAir5Property_CentralFreq.....	20

6.4.3	eComAir5Property_iDfValue	20
6.4.4	eComAir5Property_SampleRate.....	20
6.4.5	eComAir5Property_ <i>SaveRawData</i>	20
6.5	Error Code	21
6.6	Recording Resource Definition.....	21
7	Reference.....	22

Revision History

Revision	Date	Revised By	Remark
1.0.5	11/07/2014	James Lo	Support threshold.
1.0.4	09/19/2014	James Lo	Add PlayComAir5CmdList and IsComAir5CmdPlaying APIs and COMAIR5 command structure.
1.0.3	09/05/2014	James Lo	Add CalibrateComAir5Recorder and GetComAir5RecorderType APIs.
1.0.2	12/24/2013	James Lo	3 rd version
1.0.1	11/12/2013	James Lo	2 nd version
1.0.0	08/15/2013	James Lo	1 st version

1 Introduction

The COMAIR5 library is to make it easily for Android devices to interact with the devices which have Generalplus COMAIR5 chips embedded on.

This library provides following functions:

- (1) Initialize audio record unit and COMAIR5 decode function on Android platform.
- (2) Record and Decode COMAIR5 sound with build-in microphone on Android platform.
- (3) Run-time play COMAIR5 command with user's sound simultaneously.

This API specification describes all the APIs provided by this library.

2 Library Module Files

The COMAIR5 library, which is named **libGPGComAir5Lib.so**, is a shared library on Android platform. Moreover, there is a wrapper class, which is named **ComAir5Wrapper.java**, written in Java to be in charge of communicating with the COMAIR5 library and also recording/playing the sound.

2.1 Include Into an Existing Android Project

If you want to include this library into an existing Android project in Eclipse, just put **libGPGComAir5Lib.so** in the **armeabi** folder under the **libs** folder. And also include **ComAir5Wrapper.java** in the **src** folder. The folder structure is as Figure 2-1:

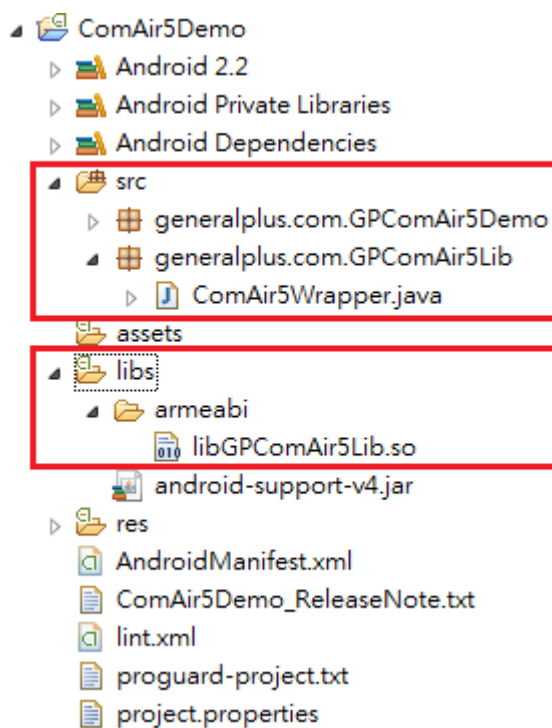


Figure 2-1 Folder Structure

2.2 Import the Demo Project

To import the demo project into an Eclipse Workspace, click “**Import...**” in the *File* menu in the toolbar. In the following, select “Existing Android Code Into Workspace”, and then enter the path of the demo project and click “Finish” in the dialog (Figure 2-2). After the above steps, the project is successfully imported.

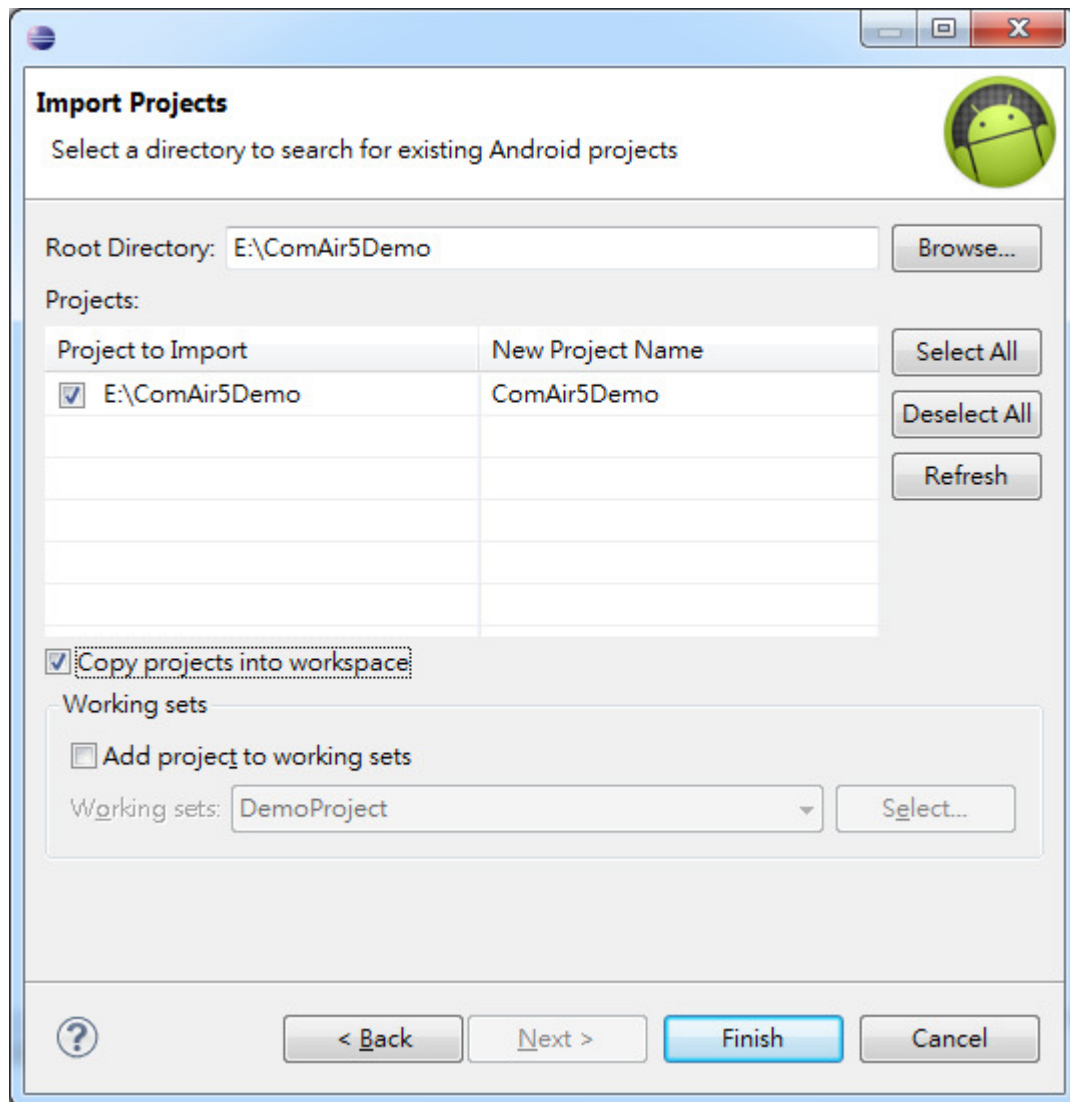


Figure 2-2 Import Existing Android Project

3 Notice for Developing App

3.1 Android Project

- 1 AndroidManifest.xml
 - 1.1 Permission RECORD_AUDIO and WRITE_EXTERNAL_STORAGE needed.
- 2 We have to define layout for size in the **res** folder.

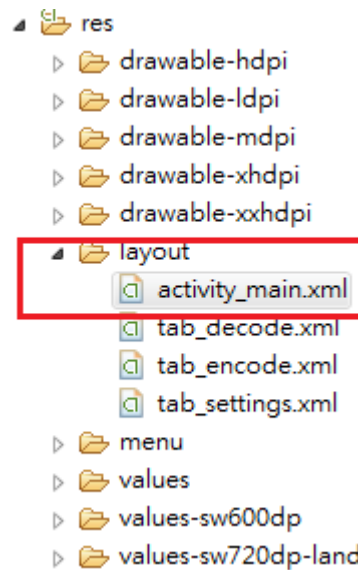


Figure 3-1 Define layout for each screen size

4 Constrains and Limitations

Constrains:

1. This library supports only Android platforms.
2. This library is compatible with Android 2.2 (API level 8) and above.
3. This library is only verified with Android devices to interact with the devices which have Generalplus COMAIR5 chips embedded on.
4. Speakers of different Android devices have different performance. Thus, users have to adjust the volume value being passed to "PlayComAir5Cmd" for each device.

5 API Calling Sequence

5.1 Decode Calling Sequence



1. The indented APIs should be called after its upper protrusive APIs were called.

6 API Specification

6.1 Data Structure

Prototype:

```
public class ComAir5Command
{
    private int      i32command; //0~63
    private float    f32Delay;
    public ComAir5Command (int iCmd, float fDelay)
    {
        i32command = iCmd;
        f32Delay = fDelay;
    }
}
```

Description:

A data structure contains COMAIR5 command.

Parameters:

i32command:

The COMAIR5 command value.

f32Delay:

The mute intervals (seconds) after playing COMAIR5 sound.

6.2 APIs Provided by the Wrapper Class

6.2.1 StartComAir5DecodeProcess

Prototype:

```
int StartComAir5DecodeProcess ()
```

Description:

Start recording and the decode process. This function calls **InitComAir5()** and **StartComAir5Decode()**. The decode process is done in a AsyncTask class. Pass in any TextView or ScrollView object in the main activity, the objects will be updated automatically if any command being decoded.

Return Value:

Return 0 (*COMAIR5_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to Section 6.5 for error code definition.

Parameters:

Remark:

6.2.2 StopComAir5DecodeProcess

Prototype:

```
int StopComAir5DecodeProcess ()
```

Description:

Start recording and the decode process. This function calls **StopComAir5Decode()** and **UnitComAir5()**.

Return Value:

Return 0 (*COMAIR5_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to Section 6.5 for error code definition.

Parameters:

Remark:

6.2.3 PlayComAirCmd

Prototype:

```
void PlayComAirCmd (int iCmd, float fVolume)
```

Description:

Play encoded sound of certain COMAIR5 command value. This function calls **PlayComAir5Cmd()**.

Return Value:

If errors occurred while generating encoded data, no sound will be played. Otherwise, the encoded sound will be played with a Runnable class.

Parameters:

[in] iCmd:

The COMAIR5 command value will be encoded.

[in] fVolume:

The level of sound volume. Only accepts the range from 0.0f to 1.0f.

Remark:

Users can play sounds simultaneously to mix them by declaring an AudioTrack instance at the same time.

6.2.4 PlayComAirCmdList

Prototype:

```
void PlayComAirCmdList (ComAirCommand[] cmdList, float fVolume)
```

Description:

Play encoded sound of a list of COMAIR5 command value. This function calls **PlayComAir5CmdList()**.

Return Value:

If errors occurred while generating encoded data, no sound will be played. Otherwise, the encoded sound will be played with a Runnable class.

Parameters:

[in] cmdList:

Specify the COMAIR5 command list

[in] fVolume:

The level of sound volume. Only accepts the range from 0.0f to 1.0f.

Remark:

Users can play sounds simultaneously to mix them by declaring an AudioTrack instance at the same time.

6.2.5 IsComAir5CmdPlaying

Prototype:

boolean IsComAir5CmdPlaying ()

Description:

Return if the encoded COMAIR5 sound is playing.

Return Value:

Return true if the encoded sound is playing. Otherwise return false.

Parameters:

Remark:

6.2.6 CalibrateComAir5Recorder

Prototype:

void CalibrateComAir5Recorder ()

Description:

On Android platform, the AudioRecord has different performances. So we will calibrate the correct parameter to initialize it. This API can auto test different parameters and get a correct parameter for your Android device.

Please refer to Section 6.6 for recording resource definition.

Return Value:

Parameters:

Remark:

6.2.7 GetComAirRecorderResourceType

Prototype:

int GetComAir5RecorderType ()

Description:

Return the audio resource value which can be used for your Android device.

Return Value:

Return the audio resource value of AudioRecord.

Parameters:

Remark:

6.3 Native APIs

6.3.1 InitComAir5

Prototype:

```
int InitComAir5 ()
```

Description:

Initialize COMAIR5 Audio unit for decoding.

Return Value:

Return 0 (*COMAIR_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to Section 6.5 for error code definition.

Parameters:

Remark:

This API should be called before any other ones.

6.3.2 UnitComAir5

Prototype:

```
int UnitComAir5 ()
```

Description:

Finalize the COMAIR5 Audio unit.

Return Value:

Return 0 (*COMAIR_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to Section 6.5 for error code definition.

Parameters:

Remark:

This API should be called before exit program.

6.3.3 StartComAir5Decode

Prototype:

```
int StartComAir5Decode ()
```

Description:

Start to decode COMAIR5 sound.

Return Value:

Return 0 (*COMAIR_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to Section 6.5 for error code definition.

Parameters:

Remark:

6.3.4 StopComAir5Decode

Prototype:

`int StopComAir5Decode ()`

Description:

Stop decoding COMAIR5 sound.

Return Value:

Return 0 (*COMAIR_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to Section 6.5 for error code definition.

Parameters:

Remark:

6.3.5 SetComAir5Property

Prototype:

`int SetComAir5Property (int eTarget, int eProperty, Object objectValue)`

Description:

Set COMAIR5 property. This function can set encode and decode COMAIR5 property separately.

Return Value:

Return 0 (*COMAIR_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to section 6.5 for error code definition.

Parameters:

[in] eTarget:

Specify target to set COMAIR5 property.
eComAir5PropertyTarget_Encode is for encode only;
eComAir5PropertyTarget_Decode is for decode only.

[in] eProperty:

The COMAIR5 property type. Please refer to Section 6.4 for property type.

[in] objectValue:

Value of the property.

Remark:

6.3.6 GetComAir5Property

Prototype:

int SetComAir5Property (int eTarget, int eProperty)

Description:

Get COMAIR5 property. This function can get encode and decode COMAIR5 property separately.

Return Value:

Return 0 (*COMAIR_NOERR*) if this function succeeded. Otherwise, other value returned. Please refer to section 6.5 for error code definition.

Parameters:

[in] eTarget:

Specify target to set COMAIR5 property. eComAir5PropertyTarget_Encode is for encode only; eComAir5PropertyTarget_Decode is for decode only.

[in] eProperty:

The COMAIR5 property type. Please refer to Section 6.4 for property type.

Remark:

6.3.7 ComAir5Decode

Prototype:

int ComAir5Decode (short[] shBuffer)

Description:

This function will pass the recorded sound to COMAIR5 library to decode.

Return Value:

If no COMAIR5 command is decoded, the return value is -1. Otherwise, the decoded command will be returned.

Parameters:

[in] shBuffer:

Specify the recorded data which is stored in a short array.

Remark:

6.3.8 PlayComAir5Cmd

Prototype:

`byte[] PlayComAir5Cmd (int i32command, float SoundVolume)`

Description:

This function will generate the encoded sound of certain COMAIR5 command and return the binary data with wave file format.

Return Value:

If there is any error, a byte array having only 1 element, which is the error code value, will be returned. Otherwise, the encoded data with wave file format will be returned. Please refer to Section 6.5 for error code definition.

Parameters:

[in] i32Command:

Specify the COMAIR5 command value to be encoded.

[in] SoundVolume:

The level of sound volume only accepts the range from 0 to 1.0.

Remark:

6.3.9 PlayComAir5CmdList

Prototype:

`byte[] PlayComAir5CmdList (int i32CommandCount, ComAir5Command[] pCommandList);`

Description:

This function will generate the encoded sound of a list of COMAIR5 command and return the binary data.

Return Value:

If there exists any error, a byte array having only 1 element, which is the error code value, will be returned. Otherwise, the encoded data with wave file format will be returned. Please refer to Section 6.5 for error code definition.

Parameters:

[in] i32CommandCount:

Specify number of the COMAIR5 command in the list.

[in] pCommandList:

Specify the COMAIR5 command list.

Remark:

6.4 COMAIR5 Property Type

Property Type	Target	Action	Function	Value Type
<i>eComAir5Property_RegCode</i>	E/D	S	Register code.	String
<i>eComAir5Property_CentralFreq</i>	E/D	S/G	Central frequency.	Int
<i>eComAir5Property_iDfValue</i>	E/D	S/G	iDf value.	Int
<i>eComAir5Property_SampleRate</i>	E/D	S/G	Sample Rate	Int
<i>eComAir5Property_SaveRawData</i>	D	S	Save Raw Data	Int
<i>eComAir5Property_Threshold</i>	D	S/G	Decode threshold.	Int

B: Both, D: Decode, E: Encode, S: Set, G: Get

6.4.1 eComAir5Property_RegCode

Set register code for Encoding/Decoding COMAIR5 command.

6.4.2 eComAir5Property_CentralFreq

Set/Get COMAIR5 Decode/Encode central frequency. The suitable central frequency is 17,000Hz.

6.4.3 eComAir5Property_iDfValue

Set/Get COMAIR5 frequency shift value. This is for advanced user only. The suitable value is 200.

6.4.4 eComAir5Property_SampleRate

Set/Get COMAIR5 sample rate value. This is for advanced user only. The suitable value is 48000 (48K Hz).

6.4.5 eComAir5Property_SaveRawData

Set to 1 if recording raw data needed, else set to 0. Default value is 0.

6.4.6 eComAir5Property_Threshold

Set the decode threshold. Default is 12.

6.5 Error Code

Symbol Name	Value	Description
<i>COMAIR5_NOERR</i>	0	No error. API works normally.
<i>COMAIR5_AUDIONOTINIT</i>	-1	Audio not initialize.
<i>COMAIR5_AUDIOUINTFAILED</i>	-2	Audio unit failed.
<i>COMAIR5_ENABLEIORECFAILED</i>	-3	Enable Record failed.
<i>COMAIR5_PROPERTYNOTFOUND</i>	-4	No such property
<i>COMAIR5_PROPERTYOPERATIONFAILED</i>	-5	Set property failed.

6.6 Recording Resource Definition

Symbol Name	Value	Description
<i>eReocorderResourceType_Default</i>	0x00	Default audio source.
<i>eReocorderResourceType_MIC</i>	0x01	Microphone audio source.
<i>eReocorderResourceType_VOICE_UPLINK</i>	0x02	Voice call uplink (Tx) audio source.
<i>eReocorderResourceType_VOICE_DOWNLINK</i>	0x03	Voice call downlink (Rx) audio source.
<i>eReocorderResourceType_VOICE_CALL</i>	0x04	Voice call uplink + downlink audio source.
<i>eReocorderResourceType_CAMCORDE R</i>	0x05	Microphone audio source with same orientation as camera if available, the main device microphone otherwise.
<i>eReocorderResourceType_VOICE_RECOGNITION</i>	0x06	Microphone audio source tuned for voice recognition if available, behaves like DEFAULT otherwise.

7 Reference

Android Developer

<http://developer.android.com/training/index.html>