AliTV Game device SDK Manual

Version：V1.05

Created：2013-12-14

Modified：2015-01-12

Modification History

No.	Version	Content	Date	Author
1	0.50	create the document	2013-12-14	Jack
2	0.60	1. delete the setOnRawDataListener API from EXDeviceManager class; 2. modify RawData class, rename mDeviceMark to mDeviceId, delete mRawDataLength; 3. delete getKeyList API from EXDevice class; 4. Add resetAMousePos API to EXDevice class	2013-12-23	Jack
3	0.61	Add three system event api to EXDevice class： sendSysKeyEvent, sendSysMouseEvent, sendSysSTouchEvent	2013-12-26	Jack
4	0.62	1. Add the api for getting Joystick and sensor data to EXDevice class： getJMotionValues, getMSensorValues 2. Add the api for setting host device to EXDeviceManager class: setHostDeviceId 3. Open the Log switch of Global class	2013-12-27	Jack
5	0.63	Add the api of “system event mode” switch to EXDeviceManager class： setEnableSysEventMode	2013-12-30	Jack
6	0.64	Add the api for getting EXDevice instance by device id to EXDeviceManager class： EXDevice getDevice(int deviceId)	2014-1-3	Jack
7	0.65	Add the api for getting host device id to EXDeviceManager class: getHostDeviceId	2014-1-8	Jack
8	0.70	Add the BaseActivity class to SDK, it could be as the parent class of app’s activity	2014-1-9	Jack
9	0.71	writeUserDefinedData api: user defined data 4 bytes to 6 bytes；	2014-1-13	Jack
10	0.72	libremote_client_sdk.so rename to：libexdeviceservicesdk.so	2014-1-16	Jack

No.	Version	Content	Date	Author
11	0.80	1. add setExpectedDeviceProperty api to EXDeviceManager calss； 2. add some writing back api to EXDevice calss: setOrientationState，setEnableSysMouse 3. add the api to get device type to EXDevice class：getDevicetType 4. add mid key and value to AMouseEvent.AMouseData 5. add return schematic diagram	2014-1-20	Jack
12	0.81	add isHostDeviceModeEnabled to EXDeviceManager	2014-1-23	Jack
13	0.82	add the api for judge features of device to EXDevice class： boolean hasFeatures(long features)	2014-2-13	Jack
14	0.83	1. add the api for getting sdk init state to EXDeviceManager class： int getInitState() 2、delete isSDKInit api from BaseActivity	2014-2-14	Jack
15	0.84	add watermark to document	2014-2-27	Jack
16	1.01	add some device type definition	2014-3-27	Jack
17	1.02	add multi-point touch event api to EXDevice class	2014-3-31	Jack
18	1.04	add TV Helper type definition to Global class	2014-6-7	Jack
19	1.05	Add alipay sdk part to doc	2014-11-21	Xuke &Zhiyong
20	1.07	Add notice when use device sdk api	2015-01-12	Xuke
21	1.08	Add notice when user wants to check if the box is running yunos system. See Environment notice	2015-01-21	Xuke
22	1.10	Change the function which is used to check if the box is running yunos system.	2015-03-02	Xuke

No.	Version	Content	Date	Author

Catalog

PART I. Device SDK

1. Introduction

“AliTV game device SDK manual” is the documentation that how to use the game device SDK of alibaba TV OS. The developers could develop or adapt the apps operated by the game devices supported by AliTV OS.

The manual just describes the main APIs, others could be understand generally.

The SDK is used to make better adaption to games with multiplayers, sensor or vibration. If there is no such need in game, developers could just use android events, but it is strongly recommended to support key and gamepad event as chapter Appendix 6

2. Environment setup

The manual just for the Ali TV OS devices, and just for the JAVA development environment. Getting started as follows:

1. put the exdeviceservicesdk.jar into the ‘libs’ folder of the project, right-click on the jar, and select Build Path -> Add to Build Path；

2. put the libexdeviceservicesdk.so into the ‘libs/armeabi/‘ directory；

3. add the permission into the AndroidManifest.xml file of the project：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

OK, now you can use the APIs of exdeviceservicesdk.jar.

Notice:

1. If developers uses the SDK API, in activity function OnKeyDown or dispatchKeyEvent should be return true when keycodes are between KEYCODE_BUTTON_A~KEYCODE_BUTTON_MODE or between KEYCODE_BUTTON_1~KEYCODE_BUTTON_16

2. If developer want to check if the box is running yunos system，use followed function:

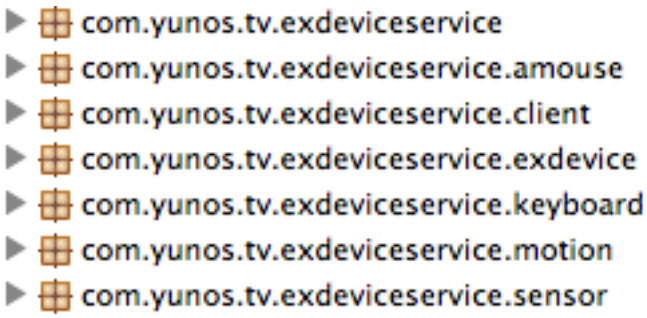
```
static boolean isYunOSSystem() {
    try {
        Class<?> sp = Class.forName("android.os.SystemProperties");
        Method m = sp.getDeclaredMethod("get", String.class);
        if (m.invoke(sp, "ro.yunos.product.chip") != null || m. invoke(sp, "ro.yunos.hardware") != null)
            return true;
    } catch (Exception e) { }

    return false;
}
```

3. SDK API

3.1. Jar package

The Jar structure as follows：

Most of important APIs are in the .client sub package, especially int the EXDeviceManger class and the EXDevice class that will be introduced later. A game device is called as EXDevice, and the devices manager is called as EXDeviceManager.



The .clent sub package also includes five event listeners as: OnAMouseListener, OnJMotionListener etc. Other sub package as .amouse, .exdevice, .keyboard, .motion, .sensor define the five events.

3.2. IEvent class

IEvent class is the parent class of the listener event, but the developer don't need to use it generally, the only thing needed to care is the both API that are used by the sub classes:

get the device ID：

public int getDeviceId()			
params	**data type**	**Meaning**	**remark**
none			
return			
int		the device ID	the identifier of the game device

get the Raw data：

public RawData getRawData()			
params	**data type**	**Meaning**	**remark**
none			
return			
RawData		the raw byte data of the event，see the RawData class	defined in: com.yunos.tv.exdeviceservice.exdevice

3.3. Global class

Need to care about something of com.yunos.tv.exdeviceservice.Global：

1) the game device type:

```
public static final int DEV_SINGLE_HAND_HANDLE = 0;
public static final int DEV_PAIR_HAND_HANDLE = 1;
public static final int DEV_DANCE_MAT = 2;
public static final int DEV_YOGA_BLANKET = 3;
public static final int DEV_STEERING_WHEEL = 4;
public static final int DEV_TOY_GUN = 5;
public static final int DEV_SOMATOSENSORY_STICK = 6;
public static final int DEV_ANALOG_GUITAR = 7;

public static final int DEV_TV_HELPER = 32;
```

2) the features value defined for devices：

One EXDevice could include multi features，one feature can be understood as a hardware module. You can get the features of device by the API of EXDevice instance. The features are defined as follows：

```
public static final long DEV_FEA_KEYBOARD = 0x0001;
public static final long DEV_FEA_AMOUSE = 0x0002;
public static final long DEV_FEA_TOUCHPAD = 0x0004;
public static final long DEV_FEA_MOTIONSTICK = 0x0008;
public static final long DEV_FEA_MSENSOR = 0x0010;
public static final long DEV_FEA_VIBRATOR = 0x0020;
public static final long DEV_FEA_MIC = 0x0040;
```

The features of one device are combined by feature bitwise OR.

3) Vibrator type definition：

```
public static final int VIBRATOR_LEFT = 0x01; //left vibrator
public static final int VIBRATOR_RIGHT = 0x02; //right vibrator
```

Just like the feature, the vibrator type is also a bitwise value, as Global.VIBRATOR_LEFT | Global.VIBRATOR_RIGHT represent the both left and right vibrator. The first input param of the ‘setVibrate’ API need vibrator type.

4) Log switch

```
public static boolean ENABLE_LOGD = false;
public static boolean ENABLE_LOGI = false;
public static boolean ENABLE_LOGE = false;
```

These are the log switch in SDK, the default value is false, and the log TAG is ‘EXDeviceServiceSDK’. Developer could open the log switch when debugging. But it’s necessary to close the log when you release the app, because the log has a great influence on app’s performance.

3.4. Event listener class

SDK support to register event listener to get the various events. you can get the device data from the event.

3.4.1. Sensor event listener（OnMSensorListener）

MSensorEvent class：

This class defines six or nine axis sensor event sent by the driver (acceleration, gravity, magnetic sensor who need external support) , mainly described the event data, use the inner class MSensorData. The API for getting data are as follows:

get event data:

```
public MSensorData getEventData()
```

params	data type	Meaning	remark
none			
return			
MSensorData		the data of sensor event	you can use the API getValue(int type) in MSensorData class to get each axis's data

The APIs in MSensorData class are as follows:

get the specific axis data value:

```
public float getValue(int type)
```

params	data type	Meaning	remark
type	int	the specific axis type	the axis types are defined in MSensorEvent class, as follows: public static final int ACCE_AXIS_X = 0; public static final int ACCE_AXIS_Y = 1; public static final int ACCE_AXIS_Z = 2; public static final int GYRO_AXIS_X = 3; public static final int GYRO_AXIS_Y = 4; public static final int GYRO_AXIS_Z = 5; public static final int MAGN_AXIS_X = 6; public static final int MAGN_AXIS_Y

public float getValue(int type)			
			= 7; public static final int MAGN_AXIS_Z = 8;
return			
float		the data value of the axis	If invalid data, just return： float MSENSOR_INVALID_DATA = 3.4E38F. The ACCE axis data unit is g (the gravity acceleration)，and the GYRO axis data is Angular velocity value.

judge the axis value is valid:

public static boolean isValidValue(float v)			
params	**data type**	**Meaning**	**remark**
v	float	the data value of an axis	
return			
boolean		true：valid false：invalid	

OnMSensorListener：

```
public interface OnMSensorListener {
      public void onMSensor(MSensorEvent event);
}
```

3.4.2. Joystick event listener（OnJMotionListener）

JMotionEvent class:

This class defines eight axis joystick event sent by the driver, mainly described the event data, use the inner class JMotionData. The API for getting data are as follows:

get event data:

`public JMotionData getEventData()`			
params	**data type**	**Meaning**	**remark**
none			
return			
JMotionData		the data of joystick event	you can use the API getValue(int type) in JMotionData class to get each axis's data

The APIs in JMotionData class are as follows:

get the specific axis data value:

`public float getValue(int type)`			
params	**data type**	**Meaning**	**remark**
type	int	the specific axis type	the axis types are defined in JMotionEvent class, as follows: public static final int AXIS_LX = 0; public static final int AXIS_LY = 1; public static final int AXIS_RX = 2; public static final int AXIS_RY = 3; public static final int AXIS_LT = 4; public static final int AXIS_RT = 5; public static final int AXIS_HAT_X = 6; public static final int AXIS_HAT_Y = 7;
return			
float		the data value of the axis	If invalid data, just return: float JMOTION_INVALID_DATA = 3.4E38F and the valid value's range is [-1.0f, 1.0f]

judge the axis value is valid:

public static boolean isValidValue(float v)			
params	**data type**	**Meaning**	**remark**
v	float	the data value of an axis	
return			
boolean		true: valid false：invalid	

OnJMotionListener：

```
public interface OnJMotionListener {
      public void onJMotion(JMotionEvent event);
}
```

3.4.3. Key event listener（OnDKeyListener）

DKeyEvent class:

This class defines key event sent by the driver, mainly described the event data, use the inner class DKeyData. The API for getting data are as follows:

get event data:

public DKeyData getEventData()			
params	**data type**	**Meaning**	**remark**
none			
return			

DKeyData		the data of key event	DKeyData public members:
			public int mSize; //array valid size
			public int[] mKeyCodes; //Key value array
			public int[] mActions; //Key Action array
			The Keys' definition at the last part of the document.
			The key Actions are defined in DKeyEvent

public DKeyData getEventData()		
		class, as follows： public static final int ACTION_UNKNOWN = -1; public static final int ACTION_DOWN = 0; public static final int ACTION_UP = 1;

CombKey:

The combination key is that two or more than two different keys are pressed at the same time, you can register a combination key with any two or more than two different keys, and the SDK will check every key event for you.

There are two constructor for CombKey:

public CombKey(String id, int key1, int key2)			
params	**data type**	**Meaning**	**remark**
id	String	the combination key's identifier, must not be duplicated	must not be null, and the id.length should larger than 0
key1	int	the first key	refrence the table at the last part of the doc
key2	int	the second key	refrence the table at the last part of the doc
return			
			the keys should be different each other

public CombKey(String id, int key1, int key2, int key3)			
params	**data type**	**Meaning**	**remark**
id	String	the combination key's identifier, must not be duplicated	must not be null, and the id.length should larger than 0
key1	int	the first key	refrence the table at the last part of the doc
key2	int	the second key	refrence the table at the last part of the doc
key3	int	the third key	refrence the table at the last part of the doc
return			
			the keys should be different each other

If you want register a CombKey with more than three keys, you should use the API as follows:

```
public void addKey(int newKey)
```

params	data type	Meaning	remark
newKey	int	the key you want to added	should be different with the existed keys in the CombKey
return			
void			

get the id of the CombKey

```
public String getId()
```

params	data type	Meaning	remark
none			
return			
String		the ID of the CombKey	

CombKeyEvent：

The CombKey event is defined in the class, the data of the event is the CombKey object. You can get the CombKey object by the API:

```
public CombKey getCombKey()
```

params	data type	Meaning	remark
无			
return			
CombKey		the CombKey of the event	

OnDKeyListener:

```
public interface OnDKeyListener {
      public void onDKey(DKeyEvent event);
      public void onCombKey(CombKeyEvent event);
}
```

3.4.4. Mouse event listener（OnAMouseListener）

AMouseEvent class:

This class defines mouse event sent by the driver, mainly described the event data, use the inner class AMouseData. The API for getting data are as follows:

get the data of event:

```
public AMouseData getEventData()
```

params	data type	Meaning	remark
none			
return			

AMouseData		the data of the mouse event	The public members of AMouseData：
			public int mPosX; //X of the absolute position
			public int mPosY; //Y of the absolute position
			public int mMidValue; //the value of the mid wheel
			public int[] mKeyStates; //left, right, mid key state

The mouse key value and state are defined in AMouseEvent class，as follows：

```
    //left, right, mid key
    public static final int MOUSE_LEFT_KEY = 0;
    public static final int MOUSE_RIGHT_KEY = 1;
    public static final int  = 2;
    //the state of the key
    public static final int MOUSE_RELEASED = 0;
    public static final int MOUSE_PRESSED = 1;
```

The main API of the inner class AMouseData:

judge the data of axis valid or not:

```
public static boolean isValidValue(float v)
```

params	data type	Meaning	remark
v	float	the data of one axis	
return			

public static boolean isValidValue(float v)		
boolean	true: valid false: invalid	can be used to check each data of AMouseData

OnAMouseListener:

```
public interface OnAMouseListener {
      public void onAMouse(AMouseEvent event);
}
```

3.4.5. Device state listener（OnEXDeviceListener）

EXDeviceEvent class:

This class defines device event sent by the driver, mainly described the event data, use the inner class EXDeviceData. The API for getting data are as follows:

get the data of event

public EXDeviceData getEventData()			
params	**data type**	**Meaning**	**remark**
none			
return			

EXDeviceData		the data of the event	the public members of EXDeviceData：
			public int mConnectionState; //connection state of device
			public int mPowerPercent; //power percent of device

The connection state of device：

```
      public static final int DEV_STATE_UNKONWN = -1;
      public static final int DEV_STATE_DISCONNECTED = 0;
      public static final int DEV_STATE_CONNECTING = 1;
      public static final int DEV_STATE_CONNECTED = 2;
```

OnEXDeviceListener：

```
public interface OnEXDeviceListener {
     public void onEXDevice(EXDeviceEvent event);
}
```

3.5. EXDeviceManager class

This class is the main provider of APIs, provides the APIs such as registering event listener, acquiring devices list, SDK init and deinit etc. It's the first class be used. The listener is restricted to device's features. You can't receive the event if the device don't support the feature even if you register the listener. SDK will call the listener in the thread of you call the API EXDeviceManager.getInstance() the first time.

This class is implemented by singleton mode, getting the singleton by the API:

`public static EXDeviceManager getInstance()`			
params	**data type**	**Meaning**	**remark**
none			
return			
EXDeviceManager		return EXDeviceManager singleton	use the instance to call the other API of EXDeviceManager

Note: you should init the singleton when you first use it, and also deinit it when you don't need it. An easy way is calling the init in onResume method and calling deinit method in onPause method of your component such as Activity. Otherwise, it could make unknown exception. the init and deinit API are:

`public void init(Context appContext, InitListener listener)`			
params	**data type**	**Meaning**	**remark**
appContext	Context	the context of the app	must not be null, otherwise throw the IllegalArgumentException
listener	InitListener	init listener	must not be null， otherwise throw the IllegalArgumentException
return			
void			

InitListener, the initialisation listener：

```
public static interface InitListener {
            //invoked when init successfully
            public void onSuccess();
            /*
            invoked when failing to init or deinit invoked
            The reasons of failing to init may be:
            1. not AliTV OS or the driver is not ready
            2. not add the INTERNET permission into AndroidManifest.xml
            */
            public void onFailed();
      }
```

public void deinit()			
params	**data type**	**Meaning**	**remark**
none			call it when you pause or exit the process
return			
void			

get the init state：

public int getInitState()			
params	**data type**	**Meaning**	**remark**
none			
return			
int		the state of init	the states are defined in EXDeviceManager class: public static final int STATE_NOT_INIT = 0; public static final int STATE_INITING = 1; public static final int STATE_INITED = 2;

set the device properties you expect:

```
public void setExpectedDeviceProperty(boolean isHorizontal, long features, boolean enableSysMouse)
```

params	data type	Meaning	remark
isHorizontal	boolean	expecting the device change to horizontal mode or not	The default value is false, it means using vertical mode. If you are developing a racing game, it's suggested to use the horizontal mode, so it should be set to true. The single hand gamepad could has both horizontal and vertical mode; The double hand gamepad only has horizontal mode. Remember to reset to false before deinit the SDK
features	long	expecting the device has specific features. using bitwise OR. you need to setup the features if you want listener the events.	The features are defined in Global class. The default value is： Global.DEV_FEA_KEYBOARD |Global.DEV_FEA_MOTIONSTICK | Global.DEV_FEA_MSENSOR So, if you want listen the mouse event, you should OR Global.DEV_FEA_AMOUSE，otherwise you won't receive it.

`public void setExpectedDeviceProperty(boolean isHorizontal, long features, boolean enableSysMouse)`			
enableSysMouse	boolean	expecting to enable or disable the system mouse	If you don't want to see the system mouse cursor in some games, you could set the param to false. But not all the devices support the property. The default value is false.
return			

get devices list:

`public List<EXDevice> getDeviceList()`			
params	**data type**	**Meaning**	**remark**
none			
return			
List<EXDevice>		the List of devices	you can get device's data by device object. see the introduction of EXDevice class

get devices count：

`public int getDeviceCount()`			
params	**data type**	**Meaning**	**remark**
none			
return			
int		the count of devices	it's just equal as getDeviceList().size()

get device object by device id:

```
public EXDevice getDevice(int deviceId)
```

params	data type	Meaning	remark
deviceId	int	device ID	each device has a identifier，SDK maintains their IDs when device plugging in or plugging out
return			
EXDevice		the object of device	

register sensor event listener:

```
public void setOnMSensorListener(OnMSensorListener listener)
```

params	data type	Meaning	remark
listener	OnMSensorListener	the listener of sensor event	it’s a interface type
return			
void			

register joystick event listener:

```
public void setOnJMotionListener(OnJMotionListener listener)
```

params	data type	Meaning	remark
listener	OnJMotionListener	joystick event listener	it’s a interface type
return			
void			

register key event listener:

```
public void setOnDKeyListener(OnDKeyListener listener)
```

params	data type	Meaning	remark
listener	OnDKeyListener	key event listener	it’s a interface type
return			

`public void setOnDKeyListener(OnDKeyListener listener)`		
void		

register mouse event listener:

`public void setOnAMouseListener(OnAMouseListener listener)`			
params	**data type**	**Meaning**	**remark**
listener	OnAMouseListener	key event listener	it's a interface type
return			
void			

register device state listener:

`public void setOnEXDeviceListener(OnEXDeviceListener listener)`			
params	**data type**	**Meaning**	**remark**
listener	OnEXDeviceListener	device state listener	it's a interface type
return			
void			

The SDK support the host device mode. In this mode, the SDK only receives the host device's event. The host device mode is disabled by default, but you can also get host device, and it's the first device in the device list. The APIs are as follows:

enable or disable host device mode:

`public boolean setEnableHostDeviceMode(boolean enable)`			
params	**data type**	**Meaning**	**remark**
enable	boolean	enable or disable the host device mode	
return			
boolean		invoked successfully or not	

is host device mode enable or not:

```
public boolean isHostDeviceModeEnabled()
```

params	data type	Meaning	remark
none			
return			
boolean		the host device mode enabled or not	

set host device id

```
public boolean setHostDeviceId(int deviceId)
```

params	data type	Meaning	remark
deviceId	int	the id of the device you want to set as host device	
return			
boolean		set successfully or not	the default host device is the first device in the device list

get the host device object

```
public EXDevice getHostDevice()
```

params	data type	Meaning	remark
none			
return			
EXDevice		host device object	

get the id of the host device

```
public int getHostDevieId()
```

params	data type	Meaning	remark
none			

`public int getHostDevieId()`		
return		
int	the id of host device	

Setup the system event mode:

Enable the system event mode: the device's events won't be sent to the SDK, these events include: key event, mouse event.

Disable the system event mode: all of device's events will be sent to the SDK, and will be sent to their listeners.

This mode is disabled by default, and it's suggested that you should not enable it as possible as you can.

`public boolean setEnableSysEventMode(boolean enable)`			
params	**data type**	**Meaning**	**remark**
enable	boolean	enable or disable the system event mode	the sensor event always be sent to the SDK, whatever the system event mode enabled or not
return			
boolean		invoked successfully or not	

register the combination keys:

`public boolean registerCombKey(CombKey ck)`			
params	**data type**	**Meaning**	**remark**
ck	CombKey	the CombKey object	see the introduction about the CombKey
return			
boolean		register successfully or not	

unregister the combination keys:

public boolean unregisterCombKey(String combKeyId)			
params	**data type**	**Meaning**	**remark**
combKeyId	String	the id of the CombKey	see the introduction about the CombKey
return			
boolean		unregister successfully or not	

get the version code of SDK

public int getVersionCode()			
params	**data type**	**Meaning**	**remark**
none			
return			
int		the version code	

get the version name of SDK

public String getVersionName()			
params	**data type**	**Meaning**	**remark**
none			
return			
String		the version name	

3.6. EXDevice class

This class defines the APIs that used to operate the device, such as: get the features of device, the device's connection state, the power percent, the keys Action, the sensor data, joystick data, mouse data, set the vibrator, enable features, get the information of device etc.

get the ID of the device

`public int getDeviceId()`			
params	**data type**	**Meaning**	**remark**
none			
return			
int		the ID of the device	The device ID is the index and identifier of the device

get the device type

`public int getDevicetType()`			
params	**data type**	**Meaning**	**remark**
none			
return			
int		the type of the device	the types are defined in the Global class

get the vendor name of the device

`public String getVendorName()`			
params	**data type**	**Meaning**	**remark**
none			
return			
String		the vendor name of the device	

get the vendor ID of the device

`public int getVendorId()`			
params	**data type**	**Meaning**	**remark**
none			
return			
int		the vendor ID of the device	

get the product name of the device

```
public String getProductName()
```

params	data type	Meaning	remark
none			
return			
String		the product name of the device	

get the product ID of the device

```
public int getProductId()
```

params	data type	Meaning	remark
none			
return			
int		the product ID of the device	

get the features of the device

```
public long getDeviceFeatures()
```

params	data type	Meaning	remark
none			
return			
long		the device features	it's a OR set of the feature. The feature is defined in Global class.

check the device has some features or not

```
public boolean hasFeatures(long features)
```

params	data type	Meaning	remark

`public boolean hasFeatures(long features)`			
features	long	the features to be checked. It's a bitwise OR value	you can check multiple feature at the same time. Like：hasFeatures(Global.DEV_FEA_MSENSOR | Global.DEV_FEA_VIBRATOR) , that means to check if the device has both sensor feature and vibrator feature.
return			
boolean		has or not	

get the device connection state:

`public int getConnState()`			
params	**data type**	**Meaning**	**remark**
none			
return			
int		get the connection state	the connection states are defined in the EXDeviceEvent class

get the device power percent:

`public int getPowerPercent()`			
params	**data type**	**Meaning**	**remark**
none			
return			
int		power percent	

get the action of the specific key:

```
public int getKeyAction(int keyCode)
```

params	data type	Meaning	remark
keyCode	int	Key value	
return			
int		the action of the key	the key actions are defined in the DKeyEvent class

get the X value of the absolute position of the mouse

```
public int getAMousePosX()
```

params	data type	Meaning	remark
none			
return			
int		the x value of the position	it's a absolute position

get the Y value of the absolute position of the mouse

```
public int getAMousePosY()
```

params	data type	Meaning	remark
无			
return			
int		the y value of the position	it's a absolute position

get the wheel value of the mid key of the mouse

```
public int getAMouseMidValue()
```

params	data type	Meaning	remark
无			
return			

public int getAMouseMidValue()		
int	the value of the mid wheel	

reset the position of the mouse

public boolean resetAMousePos(int x, int y)			
params	**data type**	**Meaning**	**remark**
x	int	the x of the postion	
y	int	the y of the position	
return			
boolean		invoked successfully or not	

get the state of mouse key

public int getAMouseState(int keyFlag)			
params	**data type**	**Meaning**	**remark**
keyFlag	int	mouse key flag	defined in the AMouseEvent class: public static final int MOUSE_LEFT_KEY = 0; public static final int MOUSE_RIGHT_KEY = 1; public static final int = 2
return			
int		the state value of the mouse key	defined in the AMouseEvent class: public static final int MOUSE_RELEASED = 0; public static final int MOUSE_PRESSED = 1; public static final int MOUSE_INVALID_DATA = 0x7FFFFFFF

get the specific axis value of joystick

```
public float getJMotionValue(int valueType)
```

params	data type	Meaning	remark
valueType	int	the specific axis of joystick	defined in the JMotionEvent class
return			
float		the axis value	If the device has no joystick feature or no the specific axis, the function will return invalid data: JMotionEvent.JMOTION_INVALID_DATA

get all axises value of joystick

```
public int getJMotionValues(float[] values)
```

params	data type	Meaning	remark
values	float[]	eight axises value of joystick	should be noted that the values[i] may be the invalid data you can use JMotionData.isValidValue to check.
return			
int		the real length of values[]	SDK supports eight axises Joystick, but if the values's length is six, it will return six.

get the specific axis value of sensor

```
public float getMSensorValue(int valueType)
```

params	data type	Meaning	remark
valueType	int	the specific axis of sensor	defined in the MSensorEvent class
return			
float		the axis value	If the device has no sensor feature or no the specific axis, the function will return invalid data: MSensorEvent.MSensor_INVALID_DATA

get all axises value of sensor

```
public float getMSensorValues(float[] values)
```

`public float getMSensorValues(float[] values)`			
params	**data type**	**Meaning**	**remark**
values	float[]	nine axises value of sensor	should be noted that the values[i] may be the invalid data you can use MSensorData.isValidValue to check.
return			
int		the real length of values[]	SDK supports nine axises Joystick, but if the values's length is six, it will return six.

enable some features of the device

`public boolean setEnableFeatures(long features, boolean enable)`			
params	**data type**	**Meaning**	**remark**
features	long	bitwise OR value of feature	defined in the Global class, but just some features can be used: DEV_FEA_AMOUSE, DEV_FEA_MSENSOR, DEV_FEA_MIC
enable	boolean	enable or disable	
return			
boolean		invoked successfully or not	not all devices support

operate the vibrator

`public boolean setVibrate(int vibratorTypes, int mode, long milliSec)`			
params	**data type**	**Meaning**	**remark**

`public boolean setVibrate(int vibratorTypes, int mode, long milliSec)`			
vibratorType	int	vibrator type	Noted: a vibrator will be open or closed is decided by the value of the same index bit of the vibratorType. For example: 0x0: close all vibrators 0x1: just open the No. 1 vibrator, close others 0x2: just open the No.2 vibrator, close others 0x3: just open both No.1 and No.2 vibrator, close others 0x4: just open the No. 3 vibrator, close others
mode	int	vibrate mode	0 - 255: the vibrating strength 0：default vibrating strength 1 - 255 vibrating strength from weak to strong，1 is most weak，255 is most strong
milliSec	long	vibrate duration	the uint is millisecond（Depending on the equipment, there may be not accurate, it is generally recommended to set between 100ms and 10000ms） -1：continuously vibrate
return			
boolean		invoked successfully or not	if the device don't support the vibrator, just return false

set the device direction mode

`public boolean setOrientationState(boolean isHorizontal)`			
params	**data type**	**Meaning**	**remark**
isHorizontal	boolean	set to horizontal mode or not	true：horizontal；false：vertical
return			

public boolean setOrientationState(boolean isHorizontal)		
boolean	invoked successfully or not	It's depended on the device. The single hand gamepad may be support horizontal and vertical mode, but the double hand gamepad only support the horizontal mode. Noted: If you set some device to horizontal mode, remember to reset to vertical mode before you call the 'deinit'

set enable the system mouse or not

public boolean setEnableSysMouse(boolean enable)			
params	**data type**	**Meaning**	**remark**
enable	boolean	enable or not	true：enabale；false：disable
return			
boolean		invoked successfully or not	It's depended on the device. some device with mouse functionality sends mouse event to the system as well as the SDK, it may be disabled to send to the system by the API.

send the key event to system:

If you need to use the API, contact with us.

public boolean sendSysKeyEvent(DKeyEvent event)			
params	**data type**	**Meaning**	**remark**
event	DKeyEvent	the SDK key event	send the SDK key event as the system key event
return			
boolean		invoked successfully or not	

send the mouse event to system

If you need to use the API, contact with us.

public boolean sendSysMouseEvent(AMouseEvent event)

`public boolean sendSysMouseEvent(AMouseEvent event)`			
params	**data type**	**Meaning**	**remark**
event	AMouseEvent	the SDK mouse event	send the SDK mouse event as the system key event 件
return			
boolean		invoked successfully or not	

send the single point touch event to system

If you need to use the API, contact with us.

`public boolean sendSysSTouchEvent(STouchEvent event)`			
params	**data type**	**Meaning**	**remark**
event	STouchEvent	the SDK single point touch event	send the SDK single point event as the system single point touch event
return			
boolean		invoked successfully or not	

send the multi-point touch event to system

If you need to use the API, contact with us.

`public boolean sendSysMTouchEvent(MTouchEvent event)`			
params	**data type**	**Meaning**	**remark**
event	MTouchEvent	the SDK multi-point touch event	send the SDK multi-point event as the system multi-point touch event
return			
boolean		invoked successfully or not	

3.7. BaseActivity class

Your Activity can be extend this class. It is a helper class, and just try to simplify the use of the SDK, and mainly does two things:

1. invoked the init and deinit API, you just need to override onSuccess and onFailed to listen the result；

2. registered the event listeners, if you want to listen some event, you just need to override the relative API like on***Event：onEXDevice, onMSensor, onJMotion, onDKey, onCombKey, onAMouse；

3.7.1. SDK initialisation

the callback of init or deinit

`public void onSuccess()`			
params	**data type**	**Meaning**	**remark**
none			invoked when init successfully
return			

`public void onFailed()`			
params	**data type**	**Meaning**	**remark**
none			invoked when failing to init or after deinit
return			

3.7.2. Event listener callback

device event callback:

`public void onEXDevice(EXDeviceEvent event)`			
params	**data type**	**Meaning**	**remark**
event	EXDeviceEvent		you can get the event when device online or offline

`public void onEXDevice(EXDeviceEvent event)`		
return		

sensor event callback

`public void onMSensor(MSensorEvent event)`			
params	**data type**	**Meaning**	**remark**
event	MSensorEvent		you can get the sensor data from the event
return			

Joystick event callback：

`public void onJMotion( event)`			
params	**data type**	**Meaning**	**remark**
event			you can get the joystick data from the event
return			

key event callback

`public void onDKey(DKeyEvent event)`			
params	**data type**	**Meaning**	**remark**
event	DKeyEvent		you can get the key data from the event
return			

combination key event callback

`public void onCombKey(CombKeyEvent event)`			
params	**data type**	**Meaning**	**remark**
event	CombKeyEvent		you can get the combination key data from the event
return			

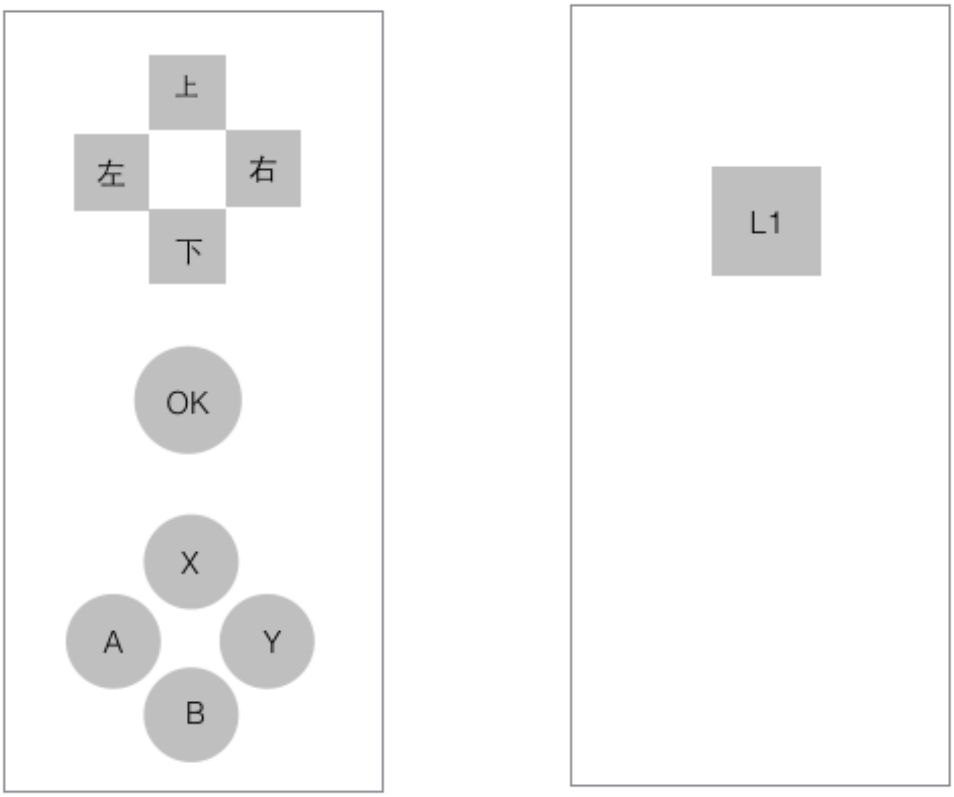
mouse event callback

`public void onAMouse(AMouseEvent event)`			
params	**data type**	**Meaning**	**remark**
event	AMouseEvent		you can get the mouse data from the event
return			

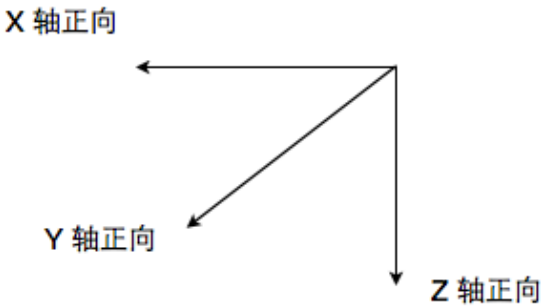
4. Gamepad Schematic diagram

4.1. Single hand gamepad

It’s a schematic diagram about the single hand gamepad and the double hand gamepad: the keys value, sensor direction.

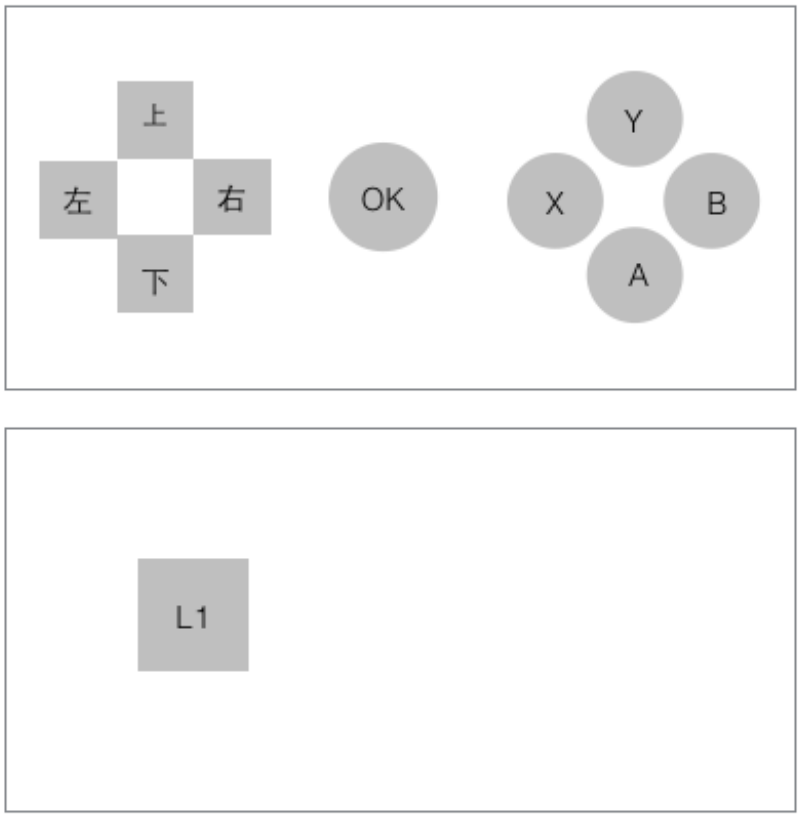


Vertical mode:

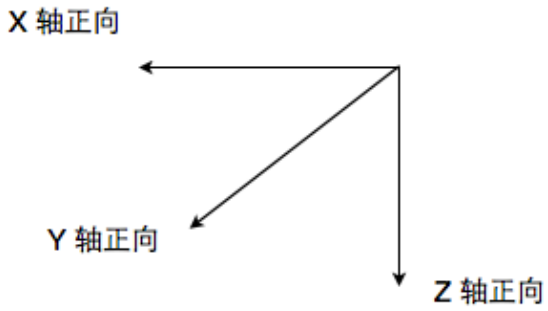


the keys schematic diagram in the vertical mode（the front and the back）

the sensor direction schematic diagram in the vertical mode



Horizontal mode:



the keys schematic diagram in the horizontal mode（the front and the back）

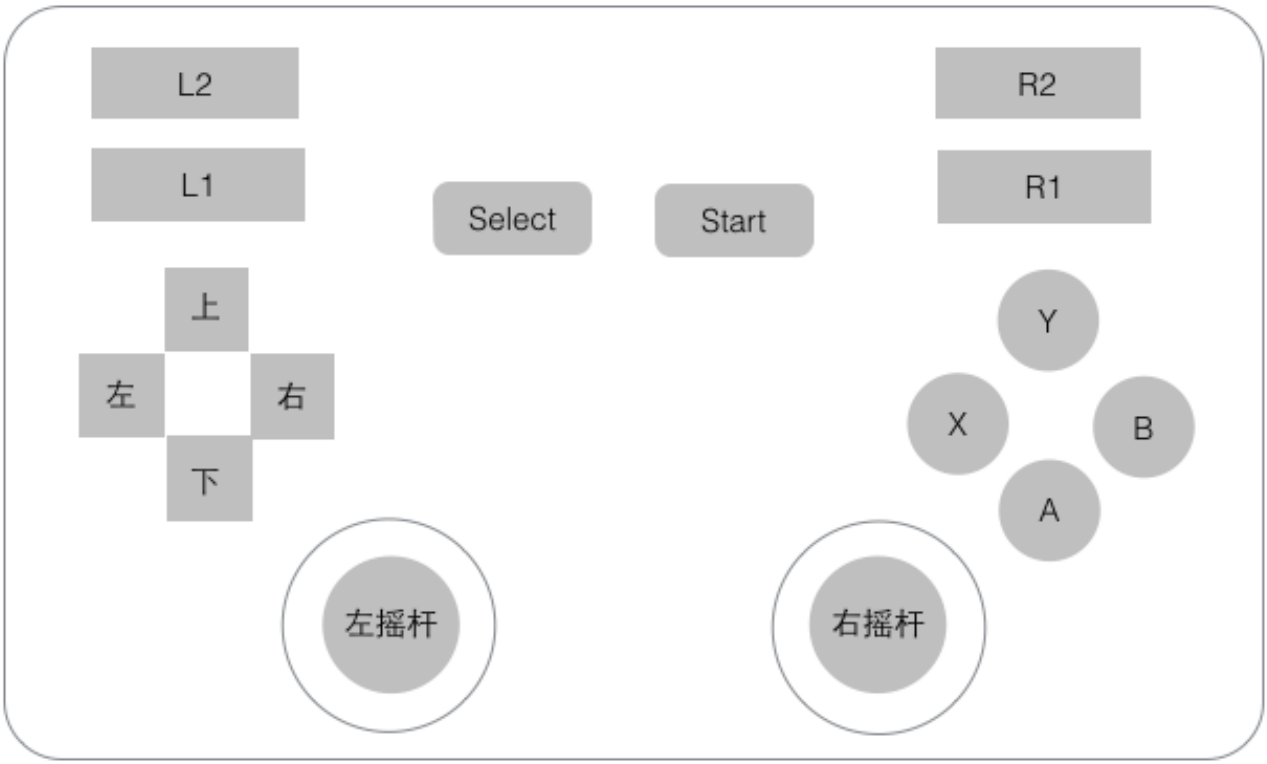
the sensor direction schematic diagram in the horizontal mode

The Keys value are defined as follows：

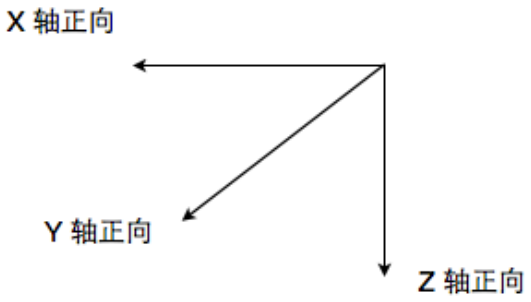
Name	Key value	remark
Left(左)	KEYCODE_DPAD_LEFT	left arrow key
Up(上)	KEYCODE_DPAD_UP	up arrow key
Right(右)	KEYCODE_DPAD_RIGHT	right arrow key
Down(下)	KEYCODE_DPAD_DOWN	down arrow key
OK	KEYCODE_DPAD_CENTER	OK key

Name	Key value	remark
A	KEYCODE_BUTTON_A	game button A key
B	KEYCODE_BUTTON_B	game button B key
X	KEYCODE_BUTTON_X	game button X key
Y	KEYCODE_BUTTON_Y	game button Y key
L1	KEYCODE_BUTTON_L1	game button L1 key

4.2. Double hand gamepad



The double hand gamepad just has horizontal mode:



the key schematic diagram

the sensor direction schematic diagram

Key value table：

名称	Key value	remark
左(Left)	KEYCODE_DPAD_LEFT	dpad left key
上(Up)	KEYCODE_DPAD_UP	dpad up key
右(Right)	KEYCODE_DPAD_RIGHT	dpad right key
下(Down)	KEYCODE_DPAD_DOWN	dpad down key
A	KEYCODE_BUTTON_A	Game: A Button key
B	KEYCODE_BUTTON_B	Game: B Button key
X	KEYCODE_BUTTON_X	Game: X Button key
Y	KEYCODE_BUTTON_Y	Game: Y Button key
L1	KEYCODE_BUTTON_L1	Game: L1 Button key
L2	KEYCODE_BUTTON_L2	Game: L2 Button key
R1	KEYCODE_BUTTON_R1	Game: R1 Button key
R2	KEYCODE_BUTTON_R2	Game: R2 Button key
Select	KEYCODE_BUTTON_SELECT	Game: Select Button key
Start	KEYCODE_BUTTON_START	Game: Start Button key
left stick key	KEYCODE_BUTTON_THUMBL	Left Thumb Button
right stick key	KEYCODE_BUTTON_THUMBR	Right Thumb Button

5. Keys Definition

All keys supported by SDK. Generally, the keys of one device are a subset of the list. The Key Codes are defined in the DKeyEvent class，the key list as follows：

Variable name	Key value	Remark
KEYCODE_UNKNOWN	0	invalid or unknown key
KEYCODE_SOFT_LEFT	1	left soft key
KEYCODE_SOFT_RIGHT	2	right soft key
KEYCODE_HOME	3	Home key
KEYCODE_BACK	4	back key
KEYCODE_CALL	5	call key
KEYCODE_ENDCALL	6	end call key
KEYCODE_0	7	number key0
KEYCODE_1	8	number key1
KEYCODE_2	9	number key2
KEYCODE_3	10	number key3
KEYCODE_4	11	number key4
KEYCODE_5	12	number key5
KEYCODE_6	13	number key6
KEYCODE_7	14	number key7
KEYCODE_8	15	number key8
KEYCODE_9	16	number key9
KEYCODE_STAR	17	* key
KEYCODE_POUND	18	# key
KEYCODE_DPAD_UP	19	DPad up key
KEYCODE_DPAD_DOWN	20	DPad down key
KEYCODE_DPAD_LEFT	21	DPad left key
KEYCODE_DPAD_RIGHT	22	DPad right key
KEYCODE_DPAD_CENTER	23	DPad center key

Variable name	Key value	Remark
KEYCODE_VOLUME_UP	24	volume up key
KEYCODE_VOLUME_DOWN	25	volumn down key
KEYCODE_POWER	26	power key
KEYCODE_CAMERA	27	Camera key
KEYCODE_CLEAR	28	Clear key
KEYCODE_A	29	character A key
KEYCODE_B	30	character B key
KEYCODE_C	31	character C key
KEYCODE_D	32	character D key
KEYCODE_E	33	character E key
KEYCODE_F	34	character F key
KEYCODE_G	35	character G key
KEYCODE_H	36	character H key
KEYCODE_I	37	character I key
KEYCODE_J	38	character J key
KEYCODE_K	39	character K key
KEYCODE_L	40	character L key
KEYCODE_M	41	character M key
KEYCODE_N	42	character N key
KEYCODE_O	43	character O key
KEYCODE_P	44	character P key
KEYCODE_Q	45	character Q key
KEYCODE_R	46	character R key
KEYCODE_S	47	character S key
KEYCODE_T	48	character T key
KEYCODE_U	49	character U key
KEYCODE_V	50	character V key

Variable name	Key value	Remark
KEYCODE_W	51	character W key
KEYCODE_X	52	character X key
KEYCODE_Y	53	character Y key
KEYCODE_Z	54	character Z key
KEYCODE_COMMA	55	, key
KEYCODE_PERIOD	56	. key
KEYCODE_ALT_LEFT	57	left ALT key
KEYCODE_ALT_RIGHT	58	right ALT key
KEYCODE_SHIFT_LEFT	59	left SHIFT key
KEYCODE_SHIFT_RIGHT	60	right SHIFT key
KEYCODE_TAB	61	TAB key
KEYCODE_SPACE	62	space key
KEYCODE_SYM	63	symbol key
KEYCODE_EXPLORER	64	Explorer key
KEYCODE_ENVELOPE	65	Envelope key
KEYCODE_ENTER	66	enter key
KEYCODE_DEL	67	DEL key
KEYCODE_GRAVE	68	` key
KEYCODE_MINUS	69	- key
KEYCODE_EQUALS	70	= key
KEYCODE_LEFT_BRACKET	71	[key
KEYCODE_RIGHT_BRACKET	72	] key
KEYCODE_BACKSLASH	73	\ key
KEYCODE_SEMICOLON	74	; key
KEYCODE_APOSTROPHE	75	‘ key
KEYCODE_SLASH	76	/ key

Variable name	Key value	Remark
KEYCODE_AT	77	@ key
KEYCODE_NUM	78	Number key
KEYCODE_HEADSETHOOK	79	Headset Hook key
KEYCODE_FOCUS	80	Camera Focus key
KEYCODE_PLUS	81	+ key
KEYCODE_MENU	82	menu key
KEYCODE_NOTIFICATION	83	Notification key
KEYCODE_SEARCH	84	Search key
KEYCODE_MEDIA_PLAY_PAUSE	85	Play/Pause media key
KEYCODE_MEDIA_STOP	86	Stop media key
KEYCODE_MEDIA_NEXT	87	Play Next media key
KEYCODE_MEDIA_PREVIOUS	88	Play Previous media key
KEYCODE_MEDIA_REWIND	89	Rewind media key
KEYCODE_MEDIA_FAST_FORWARD	90	Fast Forward media key
KEYCODE_MUTE	91	Mute key
KEYCODE_PAGE_UP	92	Page Up key
KEYCODE_PAGE_DOWN	93	Page Down key
KEYCODE_PICTSYMBOLS	94	Picture Symbols key
KEYCODE_SWITCH_CHARSET	95	Switch Charset key
KEYCODE_BUTTON_A	96	A Button key
KEYCODE_BUTTON_B	97	B Button key
KEYCODE_BUTTON_C	98	C Button key
KEYCODE_BUTTON_X	99	X Button key
KEYCODE_BUTTON_Y	100	Y Button key
KEYCODE_BUTTON_Z	101	Z Button key
KEYCODE_BUTTON_L1	102	L1 Button key

Variable name	Key value	Remark
KEYCODE_BUTTON_R1	103	R1 Button key
KEYCODE_BUTTON_L2	104	L2 Button key
KEYCODE_BUTTON_R2	105	R2 Button key
KEYCODE_BUTTON_THUMBL	106	Left Thumb Button key
KEYCODE_BUTTON_THUMBR	107	Right Thumb Button key
KEYCODE_BUTTON_START	108	Start Button key
KEYCODE_BUTTON_SELECT	109	Select Button key
KEYCODE_BUTTON_MODE	110	Mode Button key
KEYCODE_ESCAPE	111	Escape key
KEYCODE_FORWARD_DEL	112	Forward Delete key
KEYCODE_CTRL_LEFT	113	Left Control key
KEYCODE_CTRL_RIGHT	114	Right Control key
KEYCODE_CAPS_LOCK	115	Caps Lock key
KEYCODE_SCROLL_LOCK	116	Scroll Lock key
KEYCODE_META_LEFT	117	Left Meta key
KEYCODE_META_RIGHT	118	Right Meta key
KEYCODE_FUNCTION	119	Function key
KEYCODE_SYSRQ	120	System Request / Print Screen key
KEYCODE_BREAK	121	Break / Pause key
KEYCODE_MOVE_HOME	122	Home Movement key
KEYCODE_MOVE_END	123	End Movement key
KEYCODE_INSERT	124	Insert key
KEYCODE_FORWARD	125	Forward key
KEYCODE_MEDIA_PLAY	126	Play media key
KEYCODE_MEDIA_PAUSE	127	Pause media key

Variable name	Key value	Remark
KEYCODE_MEDIA_CLOSE	128	Close media key
KEYCODE_MEDIA_EJECT	129	Eject media key
KEYCODE_MEDIA_RECORD	130	Record media key
KEYCODE_F1	131	F1 key
KEYCODE_F2	132	F2 key
KEYCODE_F3	133	F3 key
KEYCODE_F4	134	F4 key
KEYCODE_F5	135	F5 key
KEYCODE_F6	136	F6 key
KEYCODE_F7	137	F7 key
KEYCODE_F8	138	F8 key
KEYCODE_F9	139	F9 key
KEYCODE_F10	140	F10 key
KEYCODE_F11	141	F11 key
KEYCODE_F12	142	F12 key
KEYCODE_NUM_LOCK	143	Num Lock key
KEYCODE_NUMPAD_0	144	Numeric keypad ‘0' key
KEYCODE_NUMPAD_1	145	Numeric keypad '1' key
KEYCODE_NUMPAD_2	146	Numeric keypad '2' key
KEYCODE_NUMPAD_3	147	Numeric keypad '3' key
KEYCODE_NUMPAD_4	148	Numeric keypad '4' key
KEYCODE_NUMPAD_5	149	Numeric keypad '5' key
KEYCODE_NUMPAD_6	150	Numeric keypad '6' key
KEYCODE_NUMPAD_7	151	Numeric keypad '7' key
KEYCODE_NUMPAD_8	152	Numeric keypad '8' key

Variable name	Key value	Remark
KEYCODE_NUMPAD_9	153	Numeric keypad '9' key
KEYCODE_NUMPAD_DIVIDE	154	Numeric keypad '/' key
KEYCODE_NUMPAD_MULTIPLY	155	Numeric keypad '*' key
KEYCODE_NUMPAD_SUBTRACT	156	Numeric keypad '-' key
KEYCODE_NUMPAD_ADD	157	Numeric keypad '+' key
KEYCODE_NUMPAD_DOT	158	Numeric keypad '.' key
KEYCODE_NUMPAD_COMMA	159	Numeric keypad ',' key
KEYCODE_NUMPAD_ENTER	160	Numeric keypad Enter key
KEYCODE_NUMPAD_EQUALS	161	Numeric keypad '=' key
KEYCODE_NUMPAD_LEFT_PAREN	162	Numeric keypad '(' key
KEYCODE_NUMPAD_RIGHT_PAREN	163	Numeric keypad ')' key
KEYCODE_VOLUME_MUTE	164	Volume Mute key
KEYCODE_INFO	165	Info key
KEYCODE_CHANNEL_UP	166	Channel up key
KEYCODE_CHANNEL_DOWN	167	Channel down key
KEYCODE_ZOOM_IN	168	Zoom in key
KEYCODE_ZOOM_OUT	169	Zoom out key
KEYCODE_TV	170	TV key
KEYCODE_WINDOW	171	Window key
KEYCODE_GUIDE	172	Guide key
KEYCODE_DVR	173	DVR key
KEYCODE_BOOKMARK	174	Bookmark key
KEYCODE_CAPTIONS	175	Toggle captions key
KEYCODE_SETTINGS	176	Settings key
KEYCODE_TV_POWER	177	V power key

Variable name	Key value	Remark
KEYCODE_TV_INPUT	178	TV input key
KEYCODE_STB_POWER	179	Set-top-box power key
KEYCODE_STB_INPUT	180	Set-top-box input key
KEYCODE_AVR_POWER	181	A/V Receiver power key
KEYCODE_AVR_INPUT	182	A/V Receiver input key
KEYCODE_PROG_RED	183	Red “programmable" key
KEYCODE_PROG_GREEN	184	Green "programmable" key
KEYCODE_PROG_YELLOW	185	Yellow “programmable" key
KEYCODE_PROG_BLUE	186	Blue "programmable" key
KEYCODE_APP_SWITCH	187	App switch key
KEYCODE_BUTTON_1	188	Generic Gamepad Button #1
KEYCODE_BUTTON_2	189	Generic Gamepad Button #2
KEYCODE_BUTTON_3	190	Generic Gamepad Button #3
KEYCODE_BUTTON_4	191	Generic Gamepad Button #4
KEYCODE_BUTTON_5	192	Generic Gamepad Button #5
KEYCODE_BUTTON_6	193	Generic Gamepad Button #6
KEYCODE_BUTTON_7	194	Generic Gamepad Button #7
KEYCODE_BUTTON_8	195	Generic Gamepad Button #8
KEYCODE_BUTTON_9	196	Generic Gamepad Button #9
KEYCODE_BUTTON_10	197	Generic Gamepad Button #10
KEYCODE_BUTTON_11	198	Generic Gamepad Button #11
KEYCODE_BUTTON_12	199	Generic Gamepad Button #12
KEYCODE_BUTTON_13	200	Generic Gamepad Button #13
KEYCODE_BUTTON_14	201	Generic Gamepad Button #14
KEYCODE_BUTTON_15	202	Generic Gamepad Button #15

Variable name	Key value	Remark
KEYCODE_BUTTON_16	203	Generic Gamepad Button #16
KEYCODE_LANGUAGE_SWITCH	204	Language Switch key
KEYCODE_MANNER_MODE	205	Manner Mode key
KEYCODE_3D_MODE	206	3D Mode key
KEYCODE_CONTACTS	207	Contacts special function key
KEYCODE_CALENDAR	208	Calendar special function key
KEYCODE_MUSIC	209	Music special function key
KEYCODE_CALCULATOR	210	Calculator special function key
KEYCODE_ZENKAKU_HANKAKU	211	Japanese full-width / half-width key
KEYCODE_EISU	212	Japanese alphanumeric key
KEYCODE_MUHENKAN	213	Japanese non-conversion key
KEYCODE_HENKAN	214	Japanese conversion key
KEYCODE_KATAKANA_HIRAGANA	215	Japanese katakana / hiragana key
KEYCODE_YEN	216	Japanese Yen key
KEYCODE_RO	217	Japanese Ro key
KEYCODE_KANA	218	Japanese kana key
KEYCODE_ASSIST	219	Assist key. Launches the global assist activity
KEYCODE_BRIGHTNESS_DOWN	220	Brightness Down key
KEYCODE_BRIGHTNESS_UP	221	Brightness Up key

6. Appendix

Key& Gamepad Event map

Linux key(scan code)	Android key	comments
KEY_LEFT (105)	KEYCODE_DPAD_LEFT	Left key
KEY_UP (103)	KEYCODE_DPAD_UP	Up key
KEY_RIGHT (106)	KEYCODE_DPAD_RIGHT	Right key

Linux key(scan code)	Android key	comments
KEY_DOWN (108)	KEYCODE_DPAD_DOWN	Down key
KEY_MENU (139)	KEYCODE_MENU	Menu key
KEY_HOMEPAGE (172)	KEYCODE_HOME	Homepage key
KEY_ESC (001)	KEYCODE_BACK	Back key
KEY_ENTER (028)	KEYCODE_DPAD_CENTER	Confirm key
BTN_A (0x130)	KEYCODE_BUTTON_A	Gamepad A Button
BTN_B (0x131)	KEYCODE_BUTTON_B	Gamepad B Button
BTN_X (0x133)	KEYCODE_BUTTON_X	Gamepad X Button
BTN_Y (0x134)	KEYCODE_BUTTON_Y	Gamepad Y Button
BTN_TL (0x136)	KEYCODE_BUTTON_L1	Gamepad L1 Button
BTN_TL2 (0x138)	KEYCODE_BUTTON_L2	Gamepad L2 Button
BTN_TR (0x137)	KEYCODE_BUTTON_R1	Gamepad R1 Button
BTN_TR2 (0x139)	KEYCODE_BUTTON_R2	Gamepad R2 Button
BTN_SELECT (0x13a)	KEYCODE_BUTTON_SELECT	Gamepad Select Button
BTN_START (0x13b)	KEYCODE_BUTTON_START	Gamepad Start Button
BTN_THUMBL (0x13d)	KEYCODE_BUTTON_THUMBL	Left Thumb Button
BTN_THUMBR (0x13e)	KEYCODE_BUTTON_THUMBR	Right Thumb Button

Joystick event

Linux AXIS	Android AXIS	Comments
ABS_X (0x00)	AXIS_X	Left joystick X direction
ABS_Y (0x01)	AXIS_Y	Left joystick Y direction
ABS_Z (0x02)	AXIS_Z	Right joystick X direction
ABS_RZ (0x05)	AXIS_RZ	Right joystick Y direction
ABS_BREAK (0x0a)	AXIS_BREAK	Left trigger
ABS_GAS (0x09)	AXIS_GAS	Right trigger
ABS_HAT0X (0x10)	AXIS_HAT_X	POV key left and right
ABS_HAT0Y (0x11)	AXIS_HAT_Y	POV key up and down