Android SDK Instruction

Content

Brief introduction

This SDK realizes the operation of printer, magnetic stripe card, cash drawer, LCD according to the methods specified in UnifiedPOS V1.141, and the printing class POSPrinter in the SDK corresponds to CHAPTER 30 POS Printer; The magnetic card class MSR in the SDK corresponds to CHAPTER 25 MSR – Magnetic Stripe Reader; The LCD class LineDisplay in the SDK corresponds to CHAPTER 22 Line Display; The cash drawer class CashDrawer in the SDK corresponds to CHAPTER 8 Cash Drawer; The serial port class is Serialport. For details about the functions implemented by each device, see SDK Interface Instruction.

一、 Interface File

possdk.aar store files in the libs folder under Android project
minSdkVersion 21 targetSdkVersion 28
dependencies add implementation fileTree(include: ['*.aar'], dir: 'libs')

二、 SDK Interface Instruction

1. Printer

```
import upos.POSException;
import upos.POSPrinter;
import upos.POSPrinterConst;
import upos.events.StatusUpdateEvent;
import upos.events.StatusUpdateListener;
```

1) Constructor

```
POSPrinter ()
POSPrinter(Context context)//    usbmanager
```

2) Printer status value callback

```
void addStatusUpdateListener (StatusUpdateListener l)
void statusUpdateOccurred(StatusUpdateEvent e);
void removeStatusUpdateListener(StatusUpdateListener l)
```

3) Open printer

```
void open(String logicalDeviceName) throws POSException
```
Parameter:
logicalDeviceName PTR_CP_2INCH :2inch ; PTR_CP_3INCH :3inch
It will throw POSException(POS_FAILURE) when open failure.

4) Close printer

```
void close()
```

5) Cutter

void cutPaper(int percentage)
Parameter:
percentage PTR_CP_FULLCUT: full cutting;
PTR_CP_PARTIALCUT: half cutting

6) Print bitmap

void printBitmap(int station, String filePath, int width, int alignment)
Parameter:
station 0 black and write picture 1 gray picture 2 Third method
filePath path of bitmap
width Print width of bitmap
alignment alignment mode PTR_BC_LEFT Left
PTR_BC_CENTER Center
PTR_BC_RIGHT Right

void printMemoryBitmap(int station, byte[] imagedata, int type, int width, int alignment)
Parameter:
station 0 black and write picture 1 gray picture 2 Third method
imagedata bitmap's data
type PTR_BMT_BMP PTR_BMT_JPEG
width bitmap's width
alignment alignment mode PTR_BC_LEFT Left
PTR_BC_CENTER Center
PTR_BC_RIGHT Right

void printBitmap(int station,Bitmap bmp, int width, int alignment)
Parameter:
station 0 black and write picture 1 gray picture 2 Third method
bmp bitmap
width bmp's width
alignment alignment mode PTR_BC_LEFT Left
PTR_BC_CENTER Center
PTR_BC_RIGHT Right

7) Send ESC data

void printNormal(int iStation, String strData)
Parameter:
iStation ignore

strData ESC data

void printNormal(int iStation, byte[] bData)
Parameter:
iStation ignore
bData ESC data bytes

8) Print barcode

void printBarCode(int station, String strData, int symbology, int height, int width, int alignment, int textPosition)
Parameter:
station ignore
strData barcode data
symbology barcode type PTR_BCS_UPCA, PTR_BCS_UPCE…PTR_BCS_***
height barcode height 1<=height<=255
width barcode width 2<=width<=6
alignment alignment mode PTR_BC_LEFT PTR_BC_CENTER PTR_BC_RIGHT
textPosition text position PTR_BC_TEXT_NONE: not printing
PTR_BC_TEXT_ABOVE: top of barcode
PTR_BC_TEXT_BELOW: bottom of barcode
PTR_BC_TEXT_BOTH: print both top and bottom

9) Open drawer

void openDrawer()

2. Serial port

import upos.POSException;
import upos.Serialport;
import upos.events.DataEvent;
import upos.events.DataReceiveListener;

1) Constructor

Serialport ()

2) Serial port data callback

void addDataReceiveListener(DataReceiveListener l)

void dataReceiveOccurred(DataEvent e)
void remeveDataReceiveListener(DataReceiveListener l)

3) Open serial port

void open(String logicalDeviceName) throws POSException
Parameter:
logicalDeviceName serial port path, Baudrate eg. “/dev/ttyS1,9600”
It will throw POSException(POS_FAILURE) when open failure.

4) Close serial port

void close()

5) Send data

void writeData(byte[] bData,int iLen)
Parameter:
bData data to send
iLen data length

6) Start and stop reading serial port data

void startReadData(boolean bStart)
Parameter:
bStart true start to read serial port thread; false stop reading data thread
Data read will be called back by listener

3. Magnetic Stripe Reader

import upos.POSException;
import upos.MSR;
import upos.events.DataEvent;
import upos.events.DataReceiveListener;

1) Constructor

MSR()

2) Magcard data callback

void addDataReceiveListener(DataReceiveListener l)
void dataReceiveOccurred(DataEvent e);
void remeveDataReceiveListener(DataReceiveListener l)

3) Open magcard device

void open(String logicalDeviceName) throws POSException
Parameter:
    logicalDeviceName ignore
It will throw POSException(POS_FAILURE) when open failure.

4) Close magcard device

void close()

5) Start to read magcard data

void authenticateDevice(byte[] buffer)
Parameter:
    buffer ignore
Magcad data read will be called back by dataReceiveOccurred

6) Stop reading magcard data thread

void deauthenticateDevice(byte[] buffer)
Parameter:
    buffer ignore

4. LCD

import upos.POSException;
import upos.LineDispaly;
import upos.LineDisplayConst ;

1) Constructor

LineDispaly()

LineDispaly(Context context)// usbmanager

2) Open LCD

void open(String logicalDeviceName) throws POSException
Parameter:

logicalDeviceName:	LINEDISPLAY_TYPE_CD6	9VFD;
	LINEDISPLAY_TYPE_CD9	32X144;
	LINEDISPLAY_TYPE_CD10	65X132;
	LINEDISPLAY_TYPE_CD8	24X128;
	LINEDISPLAY_TYPE_CD11	32X240;
	LINEDISPLAY_TYPE_COG	32X144COG;
	LINEDISPLAY_TYPE_CD13	6+8;
	LINEDISPLAY_TYPE_CD13COG	6+8 COG;
	LINEDISPLAY_TYPE_CD16	64X132CD16;
	LINEDISPLAY_TYPE_CD17	32X144CD17;
	LINEDISPLAY_TYPE_CD18	160*384 CD6;
	LINEDISPLAY_TYPE_CD19	160*320

It will throw POSException(POS_FAILURE) when open failure.
160*384 CD6 eg. logicalDeviceName = LINEDISPLAY_TYPE_CD18 +“,/dev/ttyS0”;
logicalDeviceName = LINEDISPLAY_TYPE_CD18;//usb connection
32X144COG eg. logicalDeviceName = LINEDISPLAY_TYPE_COG;

3) Close LCD

void close()

4) Set LCD display parameter

void setDescriptor(int iDescriptor,int iValue)
Parameter:

iDescriptor	LINEDISPLAY_DESCRIPTOR_BACKLIGHT	open and close backlight iValue 0 close, 1 open
	LINEDISPLAY_DESCRIPTOR_CONTRAST	set contrast iValue 0-9 64X132CD16 iValue 0-64

5) Clear LCD Screen

void clearText ()

6) Display string

void displayText(String strText)
Parameter:
strText character data
Requires LCD support for displaying strings.

void displayText(byte[] buf)
Parameter:
buf bytes array
Requires LCD support for displaying strings.

void displayText(String strText,int iFontSize)
Parameter:
strText character data
iFontSize font size

void displayText(String strText,int iFontSize,int iWidth)
Parameter:
strText strText character data
iFontSize Font size
iWidth Width of strText (iFontSize *8/15*strText.length()+20- iFontSize)
When iWidth is greater than the LCD width, it will be scrolled on screen

void displayText(String strText,int iFontSize,int iWidth,String strStatic,int iFontSize1,int iPosY)
Parameter: strText character data
iFontSize Font size
iWidth Width of strText (iFontSize *8/15*strText.length()+20- iFontSize)
strStatic Static display string
iFontSize1 Font size of strStatic
iPosY Vertical distance from origin
When iWidth is greater than the LCD width, it will be scrolled on screen

void displayText(String strText,int iFontSize,int iWidth,String strStatic,int iFontSize1,int iPosY,int iAlign)
Parameter: strText character data
iFontSize Font size
iWidth Width of strText (iFontSize *8/15*strText.length()+20- iFontSize)
strStatic Static display string
iFontSize1 Font size of strStatic
iPosY Vertical distance from origin
iAlign strStatic 0 Left align; 1 Center; 2 Right align;
When iWidth is greater than the LCD width, it will be scrolled on screen

7) Display bitmap

void displayBitmap(String path,int iWidth,int iAlignX,int iAlignY)
Parameter:
path bitmap path; ignore other parameter

void displayBitmap(Bitmap bmp)
Parameter:
bmp When bmp width is greater than the LCD width, it will be scrolled on screen

void displayBitmap(Bitmap bmp, Bitmap bmpStatic, int iPosY)
Parameter:
bmp When bmp width is greater than the LCD width, it will be scrolled on screen
bmpStatic Static image display
iPosY Vertical distance from origin

8) Set alignment

void setAlign(int iAlign)
Parameter:
iAlign 0 Left align; 1 Center; 2 Right align;

5. Drawer

import upos.POSException;
import upos.CashDrawer;

1) Constructor

CashDrawer ()

2) Drawer Status

void addStatusUpdateListener(CashDrawerUpdateListener l)
void statusUpdateOccurred(int status);
void removeStatusUpdateListener(CashDrawerUpdateListener l)

read status directly: int getStatus() 0:Opened 1:Closed; other:Error

3) Open drawer device

void open(String logicalDeviceName) throws POSException
Parameter:
logicalDeviceName ignore
It will throw POSException(POS_FAILURE) when open failure.

4) Close drawer device

void close()

5) Open drawer

void openDrawer ()

6. FingerPrint module

```
import aclasdriver.AclasFingerPrint;
import aclasdriver.data.FingerPrintListener;
import aclasdriver.data.St_FingerPrint;
```

1) Constructor

AclasFingerPrint();

2) Open fingerprint module

int AclasFingerPrintOpen(byte[] path,int iLen,FingerPrintListener listener)
Parameter:
path serial port address AOW “/dev/ttyMSM1”
iLen path.length
listener callback when register or recognize fingerprint
Return: 0 sucess; others failure
void onEventCallback(St_FingerPrint st);
St_FingerPrint :
m_iCmd 1: FingerPrint registered; 2: FingerPrint recognized
m_iCode See appendix 1 getReturnString()
m_strInfo See appendix 1

3) Close fingerprint module

void AclasFingerPrintClose()

4) Read FingerPrint ID Sum

int AclasFingerPrintReadNum(byte[] pwd,int[] ids);
Parameter:
pwd:The password(4bytes),it can be null(equal to 0x00,0x00,0x00,0x00)
ids: Used for storing ID number(0-50)
Return: 0 sucess; others failure

5) Read all FingerPrint ID

int AclasFingerPrintReadIds(byte[] pwd,ArrayList<Integer> ids)
Parameter:
pwd:The password(4 bytes),it can be null(equal to 0x00,0x00,0x00,0x00)
ids: store valid ID value
Return: 0 sucess ;others failure

6) Check whether FingerPrint ID is valid

boolean AclasFingerPrintIsValid(byte[] pwd,int id);
Parameter:
pwd The password(4 bytes),it can be null(equal to 0x00,0x00,0x00,0x00)
id FingerPrint ID to be checked
Return: true sucess valid ID;false invalid ID

7) Delete FingerPrint ID

int AclasFingerPrintDel(byte[] pwd,int type,int id);
Parameter:
pwd The password(4 bytes),it can be null(equal to 0x00,0x00,0x00,0x00)
Type 0 delete specified id; 1 delete all id
Id specified ID
Return: 0 sucess ;others failure

8) Register the fingerprint ID

Run in a thread
int AclasFingerPrintAutoEnroll(byte[] pwd,int type,int cnt, int id,int[] idOut);

Parameter:

pwd The password(4 bytes),it can be null(equal to 0x00,0x00,0x00,0x00)

type 0: There is no need to detect finger lifting, which registers until progress 100

1: It means that after pressing, you need to wait for your finger to lift and press again before you can register the next time

cnt The number of pressing can be set to 1~6 times

id Specify the ID number of the current registration fingerprint; -1 Spare ID is automatically assigned by the fingerprint module

idOut The ID value stored after registration is complete

Return: 0 sucess ;others failure

9) Recognize FingerPrint

Run in a thread

int AclasFingerPrintAutoIdentify(byte[] pwd, int[] idOut);

Parameter:

pwd The password(4 bytes),it can be null(equal to 0x00,0x00,0x00,0x00)

idOut: successfully recognized ID

Return: 0 sucess ;others failure

10) Cancel FingerPrint registration or identification

int AclasFingerPrintCancel(byte[] pwd);

Parameter:

pwd The password(4 bytes),it can be null(equal to 0x00,0x00,0x00,0x00)

Return: 0 sucess ;others failure

7. FiscalMemory

import aclasdriver.AclasBaseFunction;

1) Constructor

static AclasBaseFunction getInstance()

2) Get FM capability

int getFicalMemoryCapability(int[] iOut)

Parameter:

iOut to save the value of memory capability

Return 0 : sucess; <0: failure

3) Read data from FM

int readFicalMemory(int iAddress,int iRead,byte[] outData);
Parameter:
iAddress FM address(4 bytes) (iAddress%4==0)
iRead Data length to read
outData store data read
Return 0 : sucess; <0: failure

4) Write data to FM

int writeFicalMemory(int iAddress,int iWrite,byte[] inData);
Parameter:
iAddress FM address(4 bytes) (iAddress%4==0)
iWrite the length of inData
inData data to write in
Return 0 : sucess; <0: failure

5) Read FM status:

int getFicalMemoryStatus();
Return 1 : FM OK; <0: Communication failure; 0:FM error

8. RTC

import aclasdriver.AclasBaseFunction;

1) Constructor

static AclasBaseFunction getInstance();

2) Read date from RTC device

int getRTCTime(Date datetime);
Parameter:
datetime to save date read from RTC device.
Return 0 : sucess; <0: failure

3) Set date to RTC device

int setRTCTime(Date datetime);
Parameter:
datetime set date to RTC device.
Return 0 : sucess; <0: failure

4) Adjust RTC device date

Step: first adjusting and second adjusting at intervals of more than 12 hours
int setRTCFirstTime(Date datetime)
Parameter:
datetime The date of first adjusting
Return 0 : sucess; <0: failure

int setRTCSecondTime(Date datetime)
Parameter:
datetime The date of second adjusting
Return 0 : sucess; <0: failure

9. Cap detect

import aclasdriver.AclasCapDetect;

1) Constructor

AclasCapDetect();

2) Open cover opening sensor communication

int AclasCapDetectOpen(String strPath,int iBaudrate)
Parameter:
strPath Serial port address. AOW “/dev/ ttyMSM0”
iBaudrate: 9600
Return 0 : sucess; <0: failure

3) Close cover opening sensor communication

void AclasCapDetectClose()

4) Read version

String AclasReadVersion();
Return null: failure; others sucess; eg. “V1.006” Read information of cover opened
ArrayList<String> AclasCapDetectRead()
Return null/The list of cover opened information
format: "dd-MM-yyyy hh:mm:ss 0/1" 0 cover opened ,1 cover closed
eg. “02-08-2023 08:03:47 0”

5) Clear information of cover opened

int AclasCapDetectClear()
Return: 0 : sucess; <0: failure

6) Read date from device(Cover detect)

int AclasCapDetectRTCRead(Date datetime)
Parameter:
datetime to save the date read from device.
Return: 0 : sucess; <0: failure

7) Set date to device

int AclasCapDetectRTCSet(Date datetime)
Parameter:
datetime set date to device
Return: 0 : sucess; <0: failure

10. EJ

import aclasdriver.SDCard;

1) Constructor

SDCard()

2) Check the SD card status

int CheckCardStatus()
Return: 1 : lock ; 0 : unlock

3) Set password

int SdcardSetPasswd(byte[] pwd, int lock)
Parameter:
pwd password
Lock 0 set the password to the system
1 set the password to the system and SD card
Return 0 : sucess; <0: failure

4) Clear the password

int SdcardClearPasswd()
Return 0 : sucess; <0: failure

5) Lock the SD card

int SdcardLock()
set the password to the system before lock the SD card
Return 0 : sucess; <0: failure

6) Unlock the SD card

int SdcardUnLock()
set the password to the system before unlock the SD card
after unlock the SD card, system need sometimes to remount the SD card to the system
you can get a BroadcastReceiver to receiver action ACTION_MEDIA_MOUNTED, ACTION_MEDIA_UNMOUNTED,ACTION_MEDIA_REMOVED,ACTION_MEDIA_BAD_REMOVAL to find out the status of the SD card in mount system
Return 0 : sucess; <0: failure

Appendix 1

ErrorCode	note
0	Normally Complete
1	Unrecognized command
2	The command data length is illegal
3	The command field data is illegal
4	The system is too busy to execute the current

	command
5	If no request for the command was sent, the results were queried
6	The system software reports an error
7	Hardware error
8	No finger press is detected and exits due to timeout
9	An error occurred with the fingerprint extraction
10	An error occurred with the fingerprint match (the fingerprint template library is empty)
11	The fingerprint data storage space is full
12	Storage write failure
13	Storage read failure
14	The acquired fingerprint image is of poor quality
15	Fingerprint duplication
16	The acquisition area is too small, and the contact surface between the finger and the sensor is too small
17	The finger movement range is too large during the image collection
18	The finger movement range is too small during the image collection
19	The fingerprint ID is occupied
20	Module image collection failure
21	The command was forcibly interrupted
22	Fingerprint feature data does not need to be updated
23	Invalid fingerprint ID
24	Gain adjustment failure
25	Data buffer overflow
26	The sensor receives a message related to image collection in the sleep state and returns an error
28	Checksum error
34	Writing Flash failure while registering storage
254	Fingerprint matching failure
255	Other errors