

HT380K Application Program Interface (API)

Document Description

Contents

HT380K Application Program Interface (API) Document Description.....	1
1. The Development Specification Based on HT380K Equipment.....	3
1.1 The Development Specification.....	3
1.2 How to use Jb380_Common_V1.2.jar?.....	3
2. Scanning Module.....	4
2.1 Description of Management BarcodeManager API of Scanning Module.....	4
2.1.1 Acquire Management Objects of Scanning Module.....	4
2.1.2 Open Scanning Equipment.....	4
2.1.3 Close Scanning Equipment.....	4
2.1.4 Begin to Scan.....	5
2.1.5 Stop Scanning.....	5
2.1.6 Receive Data.....	5
2.1.7 Set up Call-back Interface.....	6
2.1.8 Open Continuous Scanning.....	6
2.1.9 Stop Continuous Scanning.....	6
3. PSAM Module.....	7
3.1 PsamController API Description of PSAM Module in Control Class.....	7
3.1.1 Acquire Management Object of PSAM Module.....	7
3.1.2 Open PSAM Module.....	7
3.1.4 Transmit Data.....	8
3.1.5 Receiver Data.....	8

1. The Development Specification Based on HT380K Equipment

1.1 The Development Specification

In order to make it easy for the customers to develop the application program based on HT380K, we provide you a jar library (named as Jb380_Common_V1.2.jar) to operate scanning module and Psam module.

The documents list description in it:

Jb380_Common_V1.2.jar	developed jar library
armeabi	so library file

1.2 How to use Jb380_Common_V1.2.jar?

- 1) Locate to your Android application program directory, create libs directory under application program directory, enter the libs directory and copy the armeabi folder under libs directory.
- 2) Create libs directory under application program directory, then copy Jb380_Common_V1.2.jar library file under lib directory.
- 3) Right-click root directory of the project, click the Properties and enter. Or pick on the root directory of the project, and press Alt-Enter.
- 4) Pick on Java Build Path in Properties page, pick on Libraries label and click Add JARs.
- 5) Find the Jb380_Common_V1.2.jar package under lib folder of project directory to be added, and click OK.
- 6) Right-click lib folder of project directory, select build Path, then select Use as Source Folder.

2. Scanning Module

2.1 Description of Management BarcodeManager API of Scanning Module

2.1.1 Acquire Management Objects of Scanning Module

Prototype	BarcodeManager getInstance()	
Function	Acquire BarcodeManager in control class, operate the scanning module through this class	
Parameters	Parameters	Descriptions
Return	BarcodeManager	BarcodeManager Objects
Note	Operate the scanning module through this class	

2.1.2 Open Scanning Equipment

Prototype	void Barcode_Open(Context context ,Callback callback)	
Function	Open scanning equipment	
Parameters	Parameters	Descriptions
	context	Caller context
	callback	Interface object of call-back data, see 2.1.6 for details
Return		
Note	The scanning equipment will be opened if this method is called back (this method will scan overhead electricity automatically), correct programmer configuration table should be ensured	

2.1.3 Close Scanning Equipment

Prototype	void Barcode_Close()	
Function	Close bar code equipment	
Parameters	Parameters	Descriptions
Return		
Note	This method will scan below electricity automatically	

2.1.4 Begin to Scan

Prototype	void Barcode_Start()	
Function	The light appears and the scanning begins	
Parameters	Parameters	Descriptions
Return		
Note		

2.1.5 Stop Scanning

Prototype	void Barcode_Stop()	
Function	The light disappears and the scanning stops	
Parameters	Parameters	Descriptions
Return		
Note		

2.1.6 Receive Data

Prototype	interface Callback { public void Barcode_Read(byte[] buffer, String codeId , int errorCode);}	
Function	Acquire the scanning data and scan the wrong information of the module	
Parameters	Parameters	Descriptions
	buffer	Scanning Data
	codeId	Bar Code Type
	errorCode	Wrong Information Value (0: No Error 1: Protocol Writing-in Failure)
Return		
Note	This interface needs to realize 1. data call-back method onDataReceived(byte[] buffer, String codeId , int errorCode)	

2.1.7 Set up Call-back Interface

Prototype	setCallback(Callback callback)	
Function	Set up Call-back Interface of Call-back Data	
Parameters	Parameters	Descriptions
	callback	Interface object of call-back data, see 2.1.6 for details
Return		
Note	This function is mainly adopted by different activities to use scanning module repeatedly, save performance cost, on the premise that (Barcode_Open ()) module is opened and (Barcode_Close ()) module is not closed; this method can be directly used to reset the call-back interface.	

2.1.8 Open Continuous Scanning

Prototype	void Barcode_Continue_Start(long time)	
Function	Open Continuous Scanning	
Parameters	Parameters	Descriptions
	time	Scanning Interval (ms)
Return		
Note		

2.1.9 Stop Continuous Scanning

Prototype	void Barcode_Continue_Stop()	
Function	Stop Continuous Scanning	
Parameters	Parameters	Descriptions
Return		
Note		

3. PSAM Module

3.1 PsamController API Description of PSAM Module in Control Class

3.1.1 Acquire Management Object of PSAM Module

Prototype	PsamController getInstance()	
Function	Acquire PsamController and operate Psam Module through this class	
Parameters	Parameters	Descriptions
Return	PsamController	PsamController Object
Note	Operate Psam Module through this class	

3.1.2 Open PSAM Module

Prototype	void Psam_Open()	
Function	Open PSAM Module	
Parameters	Parameters	Descriptions
Return		
Note	This method powers on the module automatically	

3.1.3 Close PSAM Module

Prototype	void Psam_Close()	
Function	Close PSAM Module	
Parameters	Parameters	Descriptions
Return		
Note	This method powers off the module automatically	

3.1.4 Transmit Data

Prototype	void Psam_Write(byte[] command)	
Function	Send data to PSAM module	
Parameters	Parameters	Descriptions
	command	Data Contents
Return		
Note		

3.1.5 Receiver Data

Prototype	byte[] Psam_Read() throws IOException	
Function	Acquire the returned data of PSAM module	
Parameters	Parameters	Descriptions
Return	byte[]	Returned Data
Note		