

Winmate® Software Application Note

Android General SDK – Barcode/RFID Control Application Note

Revision: 3.7

October 8, 2015

Contents

1.	Spec. Description	5
1.1	Introduction.....	5
1.2	System Overview	5
1.2.1	Device Support.....	6
1.3	Software Design – Barcode/Rfid control process	7
1.3.1	Get data from EditText.....	7
1.3.2	Get data from broadcast from WDC.....	7
1.3.3	Enable/ disable WDC (Winmate Data Capture)	7
1.3.4	Enable/ disable module power (Barcode or Rfid).....	7
1.3.5	Receive the function button event from android.....	8
1.3.6	Force to trigger Barcode/Rfid reader.....	8
1.3.7	Barcode trigger button press/release.....	8
1.3.8	Barcode trigger all mode.....	8
1.3.9	Set Barcode trigger mode.....	8
1.4	WDC Settings:	10
1.4.1	Reset.....	10
1.4.2	Set Sound Mode	10
1.4.3	Set Power Saving Mode	11
1.4.4	How to check WDC is alive.....	11
1.4.5	Display data on UI.....	11
1.4.6	Barcode scan button	11
1.4.7	Rfid scan button	12
1.5	Optional function:.....	13
1.5.1	Virtual keys (Home, Menu, Back, Search) enable/disable.....	13
2.	RFID Reader Options	14
2.1	RFID Settings:	15
2.2	UHF-Rfid Settings:	18
	Appendix	20
A.	Get data from broadcast from WDC	20
B.	Enable/disable WDC	22
C.	Enable/Disable Module Power (Barcode or Rfid)	26
D.	Receive the function button event.....	29
E.	Force to trigger barcode/Rfid reader.....	30

F.	Barcode trigger button press/release	31
G.	Barcode trigger all mode.....	32
H.	Set barcode trigger mode	34
I.	WDC Settings.....	37
I - 1	WDC Reset	38
I - 2	Set Sound Mode	39
I - 3	Set Power Saving Mode	40
I - 4	Check WDC alive.....	41
I - 5	Display Data on UI	42
I - 6	Disable Barcode scan button	43
I - 7	Disable RFID scan button	44
J.	Enable/disable virtual keys (Home, Menu, Back and Search)	45
K.	RFID Settings	47
K - 1	Read UID	48
K - 2	Read Block – ISO14443	49
K - 3	Read Block – ISO15693	50
K - 4	Write Block – ISO14443	51
K - 5	Write Block – ISO15693	52
K - 6	Set A/B Key to RFID reader	53
K - 7	Select A/B Key from RFID reader	54
K - 8	Set Trigger Mode.....	55
L.	UHF-RFID Settings.....	57
J - 1	Read Tag	58
J - 2	Write Tag	59
J - 3	Frequency Specification.....	60

Revision History

Revision	Author	Date	Description
1.0	Perry	2012/08/17	1. Initial draft
1.1	Perry	2012/08/21	1. Add force to trigger barcode/RFID reader
1.2	Perry	2012/08/28	1. Add 1.2.1: Device Support
1.3	Perry	2012/09/21	1. Add chapter 2: RFID reader options and Appendix D. 2. Add "RESULT" string to CONTENT broadcast in Appendix A.
1.4	Perry	2013/06/07	1. Add 1.3.3 : Enable/disable WDC 2. Add 1.3.4 : Receive the function button event.
1.5	Perry	2013/06/25	1. Insert 1.3.4 : Enable/ disable the power of Barcode or RFID
1.6	Perry	2013/8/29	1. Remove 1.3.4 Enable/disable the power of Barcode or RFID (There are some problems in this function) 2. Support E430M2
1.7	Perry	2013/10/03	1. Support C350M2
1.8	Coby	2014/3/11	1. Add Enable/disable the power of Barcode or RFID.
1.9	Willy	2014/3/17	1. Add set reader A/B Key 2. Add select reader A/B Key
2.0	Willy	2014/4/8	1. Fix RFID Read UID, Read block data, Write block data, Set A/B Key and Select A/B Key
3.0	Perry	2014/4/11	1. Add 1.4. : WDC Settings 2. Add "SetTriggerMode" in RFID Settings. 3. Modify all RFID APIs.
3.1	Perry	2014/4/15	1. Add 1.4.5 : Display data on UI
3.2	Willy	2014/5/12	1. Add UHF-RFID APIs.
3.3	Perry	2014/7/10	1. Modify comments of set A, B key. In Appendix H-6
3.4	Willy	2014/7/21	1. Add 1.4.6 : Barcode scan button 2. Add 1.4.7 : RFID scan button

Winmate

3.5	Willy	2014/10/9	1. Add 2.2 Frequency Specification 2. Add 1.3.7 Barcode trigger button press/release
3.6	Willy	2015/10/6	1. Modify 1.3.7 Barcode trigger button press/release
3.7	Willy	2015/10/8	1. Add 1.3.8 Barcode trigger all mode 2. Add 1.3.9 Set Barcode trigger mode

1. Spec. Description

1.1 Introduction

This specification describes how to control the barcode or RFID reader on the android phone. The developer must implement the application to meet this specification.

1.2 System Overview

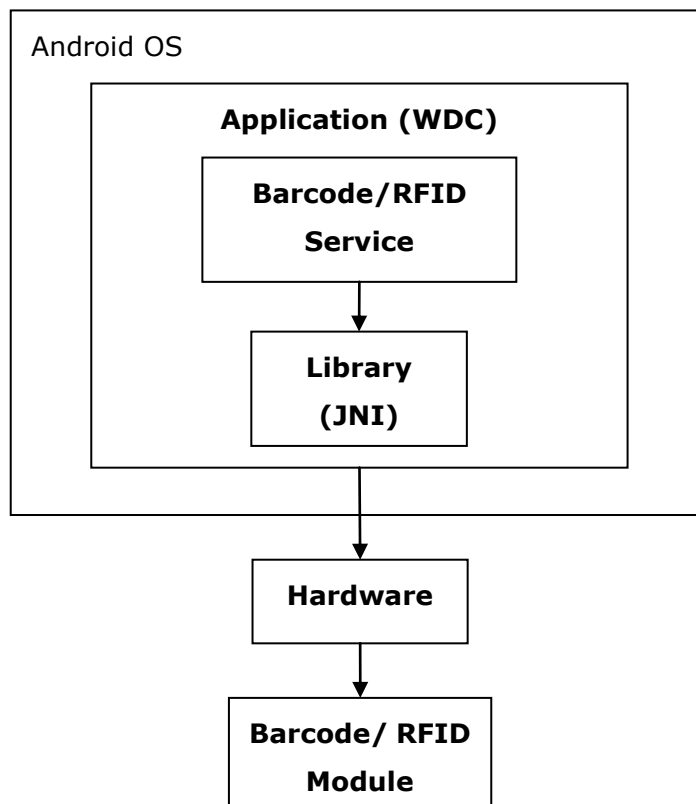
Application: Winmate Data Capture (WDC)

Service: To receive the barcode broadcast message.

Library: Access com port to controlling the barcode scanner or RFID reader.

Hardware: E430T 、E430M 、E430M2 、S430T 、C350T 、C350M2

Device Module: Opticon Barcode 2D (MDI-3100-Serial-SR), Barcode 1D (M3 CCD) or Sunion RFID (RF-521)



1.2.1 Device Support

This SDK supports the following model and device type.

<i>Model</i> <i>Device</i>	E430T	E430M E430M2	C350T C350M2	S430T
Barcode_2D (Opticon)	✓	✓	✓	✓
Barcode_1D (Opticon)	✓	✓	✓	✓
Barcode_1DL (Opticon)	✓	✓	✓	✓
RFID (Sunion)	✓	✓	✓	✓

1.3 Software Design – Barcode/RFID control process

There are two methods that can get data of barcode or RFID reader.

- 1) Get data from EditText. (see 1.3.1)
- 2) Get data from broadcast from WDC (see 1.3.2)

In addition, if you just want to receive the function button event, and then do what you want, please see 1.3.3 and 1.3.4

***Function available after WDC v.1.1.7**

1.3.1 Get data from EditText.

1. Create a new activity which contains an EditText.
2. Press the function button to triggering barcode or RFID reader.
3. Using getText() function from EditText.

Note:

If use this function, you could ignore chapter 1.3.2.

1.3.2 Get data from broadcast from WDC.

1. Create a receiver in your android application.
2. Receive broadcast from WDC and get bundle string from the intent.

See Appendix A. - [Get data from broadcast from WDC.](#)

1.3.3 Enable/ disable WDC (Winmate Data Capture)

When the mobile is booted, the barcode service will automatically run in the background. You could use the API to enable/disable WDC.

See Appendix B. - [Enable/disable WDC.](#)

1.3.4 Enable/ disable module power (Barcode or RFID)

Use the API to enable/disable module power.

See Appendix C. - [Enable/Disable Module Power \(Barcode or RFID\)](#)

1.3.5 Receive the function button event from android.

When you pressed function button, it will send a broadcast message, so you need to register a broadcast receiver in your project.

See Appendix D. – [Receive the function button event.](#)

1.3.6 Force to trigger Barcode/RFID reader.

Use the API to force triggers the barcode or RFID reader in your app.

See Appendix E. – [Force to trigger barcode/RFID reader](#)

1.3.7 Barcode trigger button press/release.

Use the API to simulate the function button behavior in your app.

See Appendix F. – [Barcode trigger button press/release](#)

1.3.8 Barcode trigger all mode.

Use the API to force trigger the barcode reader in your app, and this API support all trigger mode.

See Appendix G. – [Barcode trigger all mode](#)

1.3.9 Set Barcode trigger mode.

***Function available after WDC v.2.5.3**

Use the API to set the barcode trigger mode and trigger time in your app.

There are four trigger modes:

1. Auto: the trigger time will be ignoring.
2. Single: the trigger time will be ignoring.
3. Multiple: it also needs to set the trigger time. (Default is 2 seconds.)

4. Continuing: it also needs to set the trigger time. (Default is 2 seconds.)

See Appendix H. – [Set barcode trigger mode](#)

1.4 WDC Settings:

Use the following APIs to set WDC configuration

See Appendix I. - [WDC](#)

1.4.1 Reset

**This function available after WDC v.2.1.2 (for RFID version),
WDC v.2.1.3 (for Barcode version)*

Reset all database to default value.

Return:

WDC will return a content result "Y" when the set command is successful.
See Appendix A, how to get result from WDC.

See Appendix I-1. - [WDC Rese](#)

1.4.2 Set Sound Mode

**This function available after WDC v.2.1.2*

It will sound a good beep or vibration when module scans successes.

Sound mode:

1. Sound + Vibration
2. Sound
3. Vibration
4. None

Return:

WDC will return a content result "Y" when the set command is successful.
See Appendix A, how to get result from WDC.

See Appendix G-2. - [Set Sound Mode](#)

1.4.3 Set Power Saving Mode

**This function available after WDC v.2.1.2*

When the power saving time is over, it will stop trigger. Then it must be re-trigger RFID reader.

Power saving mode:

1. 1 (Minute)
2. 5 (Minutes)
3. 10 (Minutes)
4. None

Return:

WDC will return a content result "Y" when the set command is successful.
See Appendix A, how to get result from WDC.

See Appendix G-3. - [Set Power Saving Mode](#)

1.4.4 How to check WDC is alive

**This function available after WDC v.2.1.3*

When WDC is working, it will send a broadcast message for every 30 seconds.

Broadcast Message: `android.intent.action.WDC_ALIVE_NOTIFY`

See Appendix G-4. - [Check WDC alive](#)

1.4.5 Display data on UI

**This function available after WDC v.2.1.3*

Display data on EditText via this API.

See Appendix G-5. - [Display Data on UI](#)

1.4.6 Barcode scan button

**This function available after WDC v.2.2.2*

Winmate

Disable barcode scan button via this API.

See Appendix G-6. – [Disable Barcode scan button](#)

1.4.7 RFID scan button

***This function available after WDC v.2.2.2**

Disable rfid scan button via this API.

See Appendix G-7. – [Disable RFID scan button](#)

1.5 Optional function:

1.5.1 Virtual keys (Home, Menu, Back, Search) enable/disable

**** This function is not support for E430M, E430M2.**

You can disable the virtual keys when you using barcode function.

See Appendix H. – [Enable/disable virtual keys \(Home, Menu, Back and Search\)](#)

2. RFID Reader Options

RFID reader (Sunion) supports the tag type in the following table.

ISO 14443-A(R/W)	Mifare_One (S50)	Mifare_One (S70)		
ISO 15693(R/W)	TI HF-I Plus	TI HF-I Pro	TI HF-Standard	I-Code SLI

* The function "Read/Write Block Data" available after WDC v.1.2.3

* The function "Set/Select A/B key" available after WDC v.2.1.0

* The function "Set trigger mode" available after WDC v.2.1.2

2.1 RFID Settings:

Use the following APIs to access the RFID card.

See Appendix I: [RFID Settings](#)

1) Read UID

See Appendix I-1: [Read UID](#)

2) Read block data – ISO 14443

Read data from the card's block, the block range is 0 to 63.

See Appendix I-2: [Read Block – ISO14443](#)

3) Read block data – ISO 15693 – TI HF-I Plus – TI HF-I Pro – I-CODE2

Write data into the card's block. Please see the below table "ISO15693-TYPE" to refer the block range.

See Appendix I-3: [Read Block – ISO15693](#)

4) Write block data – ISO 14443

Write data into the card's block, the block range is 1 to 63.

Return:

WDC will return a content result "Y" when the write command is successful.
See Appendix A, how to get result from WDC.

See Appendix I-4: [Write Block – ISO14443](#)

- 5) Write block data – ISO 15693 – TI HF-I Plus
– TI HF-I Pro
– I-CODE2

Write data into the card's block. Please see the below table "ISO15693-TYPE" to refer the block range.

Return:

WDC will return a content result "Y" when the write command is successful.
See Appendix A, how to get result from WDC.

See Appendix I-5: [Write Block – ISO15693](#)

- 6) Set A/B key

Set A/B key into the RFID reader. It could store 32 keys in the RFID reader. The address range is 0 to 31.

Note:

Address 0 to 15 is A key, 16 to 31 is B key.

Return:

WDC will return a content result "Y" when the set A/B key command is successful. See Appendix A, how to get result from WDC.

See Appendix I-6: [Set A/B Key to RFID reader](#)

- 7) Select A/B key

Set A/B key into the RFID reader. It could store 32 keys in the RFID reader. The address range is 0 to 31.

Note:

Address 0 to 15 is A key, 16 to 31 is B key.

Return:

WDC will return a content result "Y" when the set A/B key command is successful. See Appendix A, how to get result from WDC.

See Appendix I-7: [Select A/B Key from RFID reader](#)

8) Set Trigger Mode

Use this function that can set trigger mode and trigger time.

There are two trigger modes:

5. Auto : the trigger time will be ignoring.
6. Single: it also needs to set the trigger time. (Default is 2 seconds.)

See Appendix I-8: [Set Trigger Mode](#)

RFID Settings

Description	Remark
Read UID	
Read block data for ISO14443-A	Block_Page = 0 - 63
Read block data for ISO15693	see below table
Write block data for ISO14443-A	Data length = 32 bytes
Write block data for ISO15693	Data length = 16 bytes
Set A/B Key value	Data length = 12 bytes A/B Key sector = 0 - 31 (0 - 15 for A Key) (16 - 31 for B Key)
Select A/B Key sector to connect to card	A/B Key sector = 0 - 31 (0 - 15 for A Key) (16 - 31 for B Key)

ISO15693 - TYPE

Description	Remark
ISO15693 - TI HF-I Plus	Block_Page = 0 - 63
ISO15693 - TI HF-I Pro	Block_Page = 0 - 11
ISO15693 - I-CODE2	Block_Page = 0 - 27

2.2 UHF-RFID Settings:

Use the following APIs to access the RFID card.

See Appendix J: [UHF-RFID Settings](#)

1) Read Tag

Read data from the area, the area range is 0 to 3.

Read length set to "0" for area is 0 ~ 2.

Read length set to any number for area is 3.

The maximum value of number depends on different tag type.

See Appendix J-1: [Read Tag](#)

2) Write Tag

Write data to the area, the area range is 0 to 2.

Return:

WDC will return a content result "Y" when the write command is successful.

See Appendix A, how to get result from WDC.

See Appendix J-2: [Write Tag](#)

3) Frequency Specification

Set frequency specification, the area range is 0 to 1.

Return:

WDC will return a content result "Y" when the setting command is successful.

See Appendix A, how to get result from WDC.

See Appendix J-3: [Frequency Specification](#)

UHF-RFID Settings

Description	Remark
Read Tag	Area = 0 - 3
Write Tag	Area = 0 - 2
Frequency Specification	Area = 0 - 1

Read Tag - Area

Description	Remark
Area 0	Password
Area 1	EPC
Area 2	TID
Area 3	User

Write Tag - Area

Description	Remark
Area 0	Password
Area 1	EPC
Area 2	User

Frequency Specification - Area

Description	Remark
Area 0	NA
Area 1	EU

Appendix

A. Get data from broadcast from WDC

Receive broadcast message from WDC to get data content.

Broadcast Message: `android.intent.action.CONTENT_NOTIFY`

Sample code:

Please add below code in your activity.

1. Create a receiver in activity and register it.

```

/*****
*   Declare broadcast message
*****/
private String ACTION_CONTENT_NOTIFY = "android.intent.action.CONTENT_NOTIFY";
/*****
*   Implement Receiver
*   get bundle: key = "CONTENT"
*               key value = data content
*   get bundle: key = "RESULT"
*               key value = writing result, if writing command success, it will get string
*               "Y".
*****/

private class DataReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(ACTION_CONTENT_NOTIFY)) {
            String content = "", result = "";
            Bundle bundle = new Bundle();
            bundle = intent.getExtras();
            content = bundle.getString("CONTENT");
            result = bundle.getString("RESULT");
            Log.w("demo", "Received Data : " + content);
            Log.w("demo", "Received Result : " + result);
        }
    }
}

```

2. Register the receiver when starting activity.

```

/*****
*   Declare class "DataReceiver"
*****/

    private DataReceiver dataReceiver = null;

/*****
*   Implement function
*   1. registerReceiver in onResume() or onCreate()
*   2. unregisterReceiver in onDestroy()
*****/

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}

@Override
protected void onResume() {
    registerReceiver();
    super.onResume();
}

@Override
protected void onDestroy() {
    unregisterReceiver();
    super.onDestroy();
}

private void registerReceiver() {
    if(dataReceiver != null) return;
    dataReceiver = new DataReceiver();
    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction(ACTION_CONTENT_NOTIFY);
    registerReceiver(dataReceiver, intentFilter);
}

private void unregisterReceiver() {
    if (dataReceiver != null) unregisterReceiver(dataReceiver);
}

```

B. Enable/disable WDC

Use this function to stop module scanning even if device was rebooted.

There are two methods to enable/ disable WDC:

1. Check/uncheck the button "WDC Enable Status" in WDC settings.
(WDC → Settings → "DC Enable Status")
2. Use the following APIs:

Options	API	Remark
Enable WDC	EnableWDC(Context context)	Enable All (Barcode and RFID)
Disable WDC	DisableWDC(Context context)	Disable All (Barcode and RFID)
Enable Barcode	EnableBarcode(Context context)	Only enable Barcode
Disable Barcode	DisableBarcode(Context context)	Only disable Barcode
Enable RFID	EnableRFID(Context context)	Only enable RFID
Disable RFID	DisableRFID(Context context)	Only disable RFID

Note:

Enable/Disable WDC (It will enable/disable barcode and RFID at the same time)

Sample code: (Enable/Disable WDC)

```

/ *****
*  _WM_GENERAL API:
*
*  EnableWDC(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void enableWDC(){
    _WM_GENERAL general = new _WM_GENERAL();
    general.EnableWDC(context);
}

/ *****
*  _WM_GENERAL API:
*
*  DisableWDC(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void disableWDC(){
    _WM_GENERAL general = new _WM_GENERAL();
    general.DisableWDC(context);
}

```


Sample code: (Enable/Disable Barcode)

```

/ *****
*  _WM_BARCODE API:
*
*  EnableBarcode(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void enableBarcode(){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.EnableBarcode(context);
}

/ *****
*  _WM_BARCODE API:
*
*  DisableBarcode(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void disableBarcode(){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.DisableBarcode(context);
}

```

Sample code: (Enable/Disable RFID)

```

/ *****
*  _WM_RFID API:
*
*  EnableRFID(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void enableRFID(){
    _WM_RFID rfid = new _WM_RFID();
    rfid.EnableRFID(context);
}

/ *****
*  _WM_RFID API:
*
*  DisableRFID(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void disableRFID(){
    _WM_RFID rfid = new _WM_RFID();
    rfid.DisableRFID(context);
}

```

C. Enable/Disable Module Power (Barcode or RFID)

Use the following APIs to enable/ disable module's power:

Options	API	Remark
Power On (Barcode)	BarPowerOn(Context context)	Enable barcode power
Power Off (Barcode)	BarPowerOff(Context context)	Disable barcode power
Power On (RFID)	RFIDPowerOn(Context context)	Enable RFID power
Power Off (RFID)	RFIDPowerOff(Context context)	Disable RFID power

Sample code: (Enable/Disable barcode power)

```

/ *****
*  _WM_BARCODE API:
*
*  BarPowerOn(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void enableBarcodePower(){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.BarPowerOn(context);
}

/ *****
*  _WM_BARCODE API:
*
*  BarPowerOff(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void disableBarcodePower (){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.BarPowerOff(context);
}

```

Sample code: (Enable/Disable RFID power)

```

/ *****
*  _WM_RFID API:
*
*  RFIDPowerOn(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void enableRFIDPower(){
    _WM_RFID rfid = new _WM_RFID();
    rfid.RFIDPowerOn(context);
}

/ *****
*  _WM_RFID API:
*
*  RFIDPowerOff(Context context)
*
*      Parameters :
*          context      The context of current state of the application/object
*
*****/

private void disableRFIDPower(){
    _WM_RFID rfid = new _WM_RFID();
    rfid.RFIDPowerOff(context);
}

```

D. Receive the function button event.

Receive the broadcast when user press function button.

Device Type	Mobile Type	Button Event (Broadcast Message)
Barcode	E430T S430T C350T C350M2	<i>android.intent.action.BARCODE_NOTIFY</i>
RFID	E430T S430T C350T C350M2	<i>android.intent.action.RFID_NOTIFY</i>
Barcode RFID	E430M E430M2	<i>android.intent.action.BARCODE_NOTIFY</i>

1) Register a broadcast receiver in the AndroidManifest.xml

2) Implement a broadcast receiver

Sample code (for Barcode type):

```

/*****
*   Register a broadcast receiver in the AndroidManifest.xml
*****/

<receiver android:name=".BarcodeDemoReceiver" >
    <intent-filter>
        <action android:name="android.intent.action.BARCODE_NOTIFY" />
        <category android:name="android.intent.category.HOME" />
    </intent-filter>
</receiver>

// ****
*   Implement a broadcast receiver
*****/

public class BarcodeDemoReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.intent.action.BARCODE_NOTIFY")) {
            // Do something...
        }
    }
}

```

E. Force to trigger barcode/RFID reader

```

/ *****
*  _WM_BARCODE API:
*
*  BarTrigger(Context context)
*
*      Parameters :
*      context      The context of current state of the application/object
*
***** /

private void Barcode_Trigger(){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.BarTrigger(context);
}

/ *****
*  _WM_RFID API:
*
*  RFIDTrigger(Context context)
*
*      Parameters :
*      context      The context of current state of the application/object
*
***** /

private void RFID_Trigger(){
    _WM_RFID rfid = new _WM_RFID();
    rfid.RFIDTrigger(context);
}

```

F. Barcode trigger button press/release

```

/ *****
*  _WM_BARCODE API:
*
*  BarButtonPress(Context context)
*
*  Parameters :
*      context      The context of current state of the application/object
*
***** /

private void Barcode_Button_Press(){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.BarButtonPress(context);
}

/ *****
*  _WM_RFID API:
*
*  BarButtonRelease(Context context)
*
*  Parameters :
*      context      The context of current state of the application/object
*
***** /

private void Barcode_Button_Release(){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.BarButtonRelease(context);
}

```


G. Barcode trigger all mode

```

/ *****
*  _WM_BARCODE API:
*
*  BarAllModeTrigger(Context context)
*
*  Parameters :
*      context      The context of current state of the application/object
*
*  Note :
*      You have to implement this function in UI.
*      Sample code will show how to implement this function at UI button.
*
*      OnTouchListener will detect the UI button touch motion, and ACTION_DOWN means user
*      touch the button, ACTION_UP means user release the button.
***** /

private void Barcode_All_Mode_Trigger(){
    _WM_BARCODE bar = new _WM_BARCODE();
    bar.BarAllModeTrigger(context);
}

```

Sample code for Barcode trigger all mode:

```
private Button mBtnTriAllMode;

mBtnTriAllMode.setOnTouchListener(new Button.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                Barcode_All_Mode_Trigger();
                break;
            case MotionEvent.ACTION_UP:
                Barcode_Button_Release();
                break;
            default:
                break;
        }
        return false;
    }
});
}
```

H. Set barcode trigger mode

```

/*****
*
*  _WM_BARCODE API:
*
*  SetTriggerMode(Context context, TriggerMode mode, TriggerTime time)
*
*  Parameters :
*
*      context      The context of current state of the application/object
*      mode          enum TriggerMode
*      time          enum TriggerTime
*
*  The type is:
*
*      public static enum TriggerMode {
*
*          Auto,
*
*          Single,
*
*          Multiple,
*
*          Continuing;
*
*      }
*
*  The time is:
*
*      public static enum TriggerTime {
*
*          Sec_1,
*
*          Sec_2,
*
*          Sec_3,
*
*          Sec_4,
*
*          Sec_5,
*
*          Sec_6,
*
*          Sec_7,
*
*          Sec_8,
*
*          Sec_9;
*
*      }
*
*  Trigger Mode:
*
*      1. _WM_BARCODE.TriggerMode.Auto
*
*      2. _WM_BARCODE.TriggerMode.Single
*
*      3. _WM_BARCODE.TriggerMode.Multiple
*
*      4. _WM_BARCODE.TriggerMode.Continuing
*
*
*  Trigger Time:
*
*      1. _WM_BARCODE.TriggerTime.Sec_1

```

Winmate

```
*      ...
*      ...
*      9. _WM_BARCODE.TriggerTime.Sec_9
*
*  Note:
*      If trigger mode is selected "Auto" or "Single", the trigger time will be ignore.
*
*      After you set trigger mode, then you have to wait 2 seconds for module initial.
*  *****/

//auto trigger
private void set_trigger_mode() {
    _WM_RFID bar = new _WM_BARCODE();
    //set trigger to auto trigger, trigger time will be ignored.
    bar.SetTriggerMode(mContext,
        _WM_BARCODE.TriggerMode.Auto, _WM_BARCODE.TriggerTime.Sec_2);

    try {Thread.sleep(2000);} catch (InterruptedException e)
    {e.printStackTrace();}
}

//single trigger
private void set_trigger_mode() {
    _WM_RFID bar = new _WM_BARCODE();
    //set trigger to single trigger, trigger time will be ignored.
    //set trigger to single trigger, trigger time = 2 seconds.
    bar.SetTriggerMode(mContext,
        _WM_BARCODE.TriggerMode.Single, _WM_BARCODE.TriggerTime.Sec_2);

    try {Thread.sleep(2000);} catch (InterruptedException e)
    {e.printStackTrace();}
}

//multiple trigger
private void set_trigger_mode() {
    _WM_RFID bar = new _WM_BARCODE();
    //set trigger to multiple trigger, trigger time = 2 seconds.
    bar.SetTriggerMode(mContext,
        _WM_BARCODE.TriggerMode.Multiple, _WM_BARCODE.TriggerTime.Sec_2);
}
```

```
        try {Thread.sleep(2000);} catch (InterruptedException e)
        {e.printStackTrace();}
    }

    //continuing trigger
    private void set_trigger_mode() {
        _WM_RFID bar = new _WM_BARCODE();
        //set trigger to continuing trigger, trigger time = 2 seconds.
        bar.SetTriggerMode(mContext,
            _WM_BARCODE.TriggerMode.Continuing, _WM_BARCODE.TriggerTime.Sec_2);

        try {Thread.sleep(2000);} catch (InterruptedException e)
        {e.printStackTrace();}
    }
```

I. WDC Settings

Use the following APIs to set WDC configuration.

I - 1 WDC Rese

I - 2 Set Sound Mode

I - 3 Set Power Saving Mode

I - 4 Check WDC alive

I - 5 Display Data on UI

Note:

*If WDC responded result is “Y”, it means setting command is successful.
See Appendix A, how to get result from WDC.*

I - 1 WDC Reset

```

/ *****
*  _WM_GENERAL API:
*
*  Reset(Context context)
*
*  Parameters :
*      context      The context of current state of the application/object
*
***** /

private void reset(){
    _WM_GENERAL general = new _WM_GENERAL();
    general.Reset(context);
}

```

I - 2 Set Sound Mode

```

/*****
*  _WM_GENERAL API:
*
*  SetSoundMode(Context context, SoundMode mode)
*
*  Parameters :
*      context      The context of current state of the application/object
*      mode         enum SoundMode
*
*  The mode is:
*      public static enum SoundMode {
*          Sound_Vibration,
*          Sound,
*          Vibration,
*          None
*      }
*
*  Sound Mode:
*      1. _WM_GENERAL.SoundMode.Sound_Vibration
*      2. _WM_GENERAL.SoundMode.Sound
*      3. _WM_GENERAL.SoundMode.Vibration
*      4. _WM_GENERAL.SoundMode.None
*
*****/

private void set_sound_mode() {
    _WM_GENERAL general = new _WM_GENERAL();
    // set sound mode to only vibration.
    general.SetSoundMode(context, _WM_GENERAL.SoundMode.Vibration);
}

```


I - 3 Set Power Saving Mode

```

/*****
 *  _WM_GENERAL API:
 *
 *  SetPowerSaving(Context context, PowerSaving mode)
 *
 *  Parameters :
 *
 *      context      The context of current state of the application/object
 *      mode         enum PowerSaving
 *
 *  The mode is:
 *
 *      public static enum PowerSaving {
 *
 *          Minute_1,
 *
 *          Minute_5,
 *
 *          Minute_10,
 *
 *          None
 *
 *      }
 *
 *  Power Saving Mode:
 *
 *      1. _WM_GENERAL.PowerSaving.Minute_1
 *      2. _WM_GENERAL.PowerSaving.Minute_5
 *      3. _WM_GENERAL.PowerSaving.Minute_10
 *      4. _WM_GENERAL.PowerSaving.None
 *
 *****/

private void set_power_saving_mode() {
    _WM_GENERAL general = new _WM_GENERAL();
    // set power saving mode to "None" (No control)
    general.SetPowerSaving(context, _WM_GENERAL.PowerSaving.None);
}

```

I - 4 Check WDC alive

```

/*****
*   Register a broadcast receiver in the AndroidManifest.xml
*****/

<receiver android:name=".WDC_Alive_Receiver" >
    <intent-filter>
        <action android:name="android.intent.action.WDC_ALIVE_NOTIFY" />
        <category android:name="android.intent.category.HOME" />
    </intent-filter>
</receiver>

// ****
*   Implement a broadcast receiver
*****/

public class WDC_Alive_Receiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.intent.action.WDC_ALIVE_NOTIFY")) {
            // Do something...
        }
    }
}

```

I - 5 Display Data on UI

```

/*****
 *  _WM_GENERAL API:
 *
 *  SetDisplayData(Context context, boolean isDisplay)
 *
 *  Parameters :
 *      context      The context of current state of the application/object
 *      isDisplay    true : display data on UI
 *                  false: Doesn't display data on UI.
 *
 *****/

//display data on UI
private void set_display_data() {
    _WM_GENERAL general = new _WM_GENERAL();
    general.SetDisplayData(context, true);
}

//doesn't display data on UI
private void set_display_data() {
    _WM_GENERAL general = new _WM_GENERAL();
    general.SetDisplayData(context, true);
}

```

I - 6 Disable Barcode scan button

```

/*****
 *  _WM_GENERAL API:
 *
 *  SetBarcodeScanButton(Context context, boolean isDisable)
 *
 *  Parameters :
 *      context      The context of current state of the application/object
 *      isDisable    true : Disable barcode scan button.
 *                  false: Enable barcode scan button.
 *
 *****/

//Disable barcode scan button
private void set_barcode_scan_button() {
    _WM_GENERAL general = new _WM_GENERAL();
    general.SetBarcodeScanButton(context, true);
}

//Enable barcode scan button
private void set_barcode_scan_button () {
    _WM_GENERAL general = new _WM_GENERAL();
    general.SetBarcodeScanButton(context, false);
}

```

I - 7 Disable RFID scan button

```

/*****
*   _WM_GENERAL API:
*
*   SetRFIDScanButton(Context context, boolean isDisable)
*
*   Parameters :
*       context      The context of current state of the application/object
*       isDisable    true : Disable rfid scan button.
*                   false: Enable rfid scan button.
*
*****/

//Disable barcode scan button
private void set_rfid_scan_button() {
    _WM_GENERAL general = new _WM_GENERAL();
    general.SetRFIDScanButton(context, true);
}

//Enable barcode scan button
private void set_rfid_scan_button () {
    _WM_GENERAL general = new _WM_GENERAL();
    general.SetRFIDScanButton(context, false);
}

```

J. Enable/disable virtual keys (Home, Menu, Back and Search)

**** This function is not support for E430M, E430M2.**

Access the node `"/sys/board_properties/touch_disable"` via JNI layer.

Write data `"0"` to enable touch panel.

Write data `"1"` to disable touch panel.

Write data `"2"` to disable virtual keys.

Note:

If you write data `"1"` then you cannot access any function on the touch screen, so you must be careful to use this function.

Sample code for Java layer:

```

/*****
 *   Declare Native Function
 *****/

    public native int TouchStatus(String path, String data);

// *****/

 *   Implement function
 *
 *   path = /sys/board_properties/touch_disable
 *   Enable touch panel => data = "0"
 *   Disable touch panel => data = "1"
 *   Disable virtual key => data = "2"
 *****/

    public void VirtualKeyStatus(int iState) {
        //touch_disable node path
        String path = "/sys/board_properties/touch_disable";
        //set touch state
        TouchStatus(path, iState);
    }

```

Sample code for JNI layer:

```

/*****
 *   Implement native function
 *
 *   path = /sys/board_properties/touch_disable
 *
 *****/
JNIEXPORT jint JNICALL Java_com_winmate_demo_main_TouchStatus (JNIEnv * env,
jobject thiz, jstring path, jstring data) {
    const char *cPath, *cData;
    if (path == NULL) return -1;
    // Convert jstring to char pointer
    cPath = (*env)->GetStringUTFChars(env, path, NULL);
    cData = (*env)->GetStringUTFChars(env, data, NULL);
    // Open node "/sys/board_properties/touch_disable"
    int fd = open(cPath, O_WRONLY | O_NOCTTY | O_NDELAY | O_NONBLOCK);
    if (fd < 0) {
        // Release char pointer
        (*env)->ReleaseStringUTFChars(env, path, cPath);
        (*env)->ReleaseStringUTFChars(env, data, cData);
        return -1;
    }
    // Write data to the node "/sys/board_properties/touch_disable",
    // Enable touch panel => data = "0"
    // Disable touch panel => data = "1"
    // Disable virtual key => data = "2"
    write(fd, cData, strlen(cData));
    // Release char pointer
    (*env)->ReleaseStringUTFChars(env, path, cPath);
    (*env)->ReleaseStringUTFChars(env, data, cData);
    close(fd);

    return 0;
}

```

K. RFID Settings

Use the following APIs to access the RFID card.

K - 1 Read UID

K - 2 Read Block – ISO14443

K - 3 Read Block – ISO15693

K - 4 Write Block – ISO14443

K - 5 Write Block – ISO15693

K - 6 Set A/B Key to RFID reader

K - 7 Select A/B Key from RFID reader

K - 8 Set Trigger Mode

Note:

*If WDC responded result is “Y”, it means setting command is successful.
See Appendix A, how to get result from WDC.*

K - 1 Read UID

```

/*****
*  _WM_RFID API:
*
*  ReadUID(Context context)
*
*  Parameters :
*      context      The context of current state of the application/object
*
*****/

private void read_uid() {
    _WM_RFID rfid = new _WM_RFID();
    rfid.ReadUID(context);
}

```

K - 2 Read Block – ISO14443

```

/*****
 *  _WM_RFID API:
 *
 *  ReadBlock_ISO14443(Context context, int blockPage)
 *
 *  Parameters :
 *      context      The context of current state of the application/object
 *      blockPage    blockPage(0 - 63)
 *
 *  Note:
 *      blockPage = 0, This block saves card ID.
 *      blockPage = 3, 7, 11, 15, 19, ..., 63, Those blocks save AB Key.
 *      Another blocks save user data. ex: 1, 2, 4, 5, 6, 8, ...
 *****/

private void read_block_iso14443() {
    _WM_RFID rfid = new _WM_RFID();
    //read data from block(20)
    rfid.ReadBlock_ISO14443(context, 20);
}

```

K - 3 Read Block – ISO15693

```

/*****
 *  _WM_RFID API:
 *
 *  ReadBlock_ISO15693(Context context, ISO15693Type type, int blockPage)
 *
 *  Parameters :
 *
 *      context      The context of current state of the application/object
 *      type         enum ISO15693Type
 *      blockPage    see Note.
 *
 *  The type is:
 *
 *      public static enum ISO15693Type {
 *
 *          TI_HF_I_Plus,
 *          TI_HF_I_Pro,
 *          I_CODE2
 *
 *      }
 *
 *  Power Saving Mode:
 *
 *      1. _WM_RFID.ISO15693Type.TI_HF_I_Plus
 *      2. _WM_RFID.ISO15693Type.TI_HF_I_Pro
 *      3. _WM_RFID.ISO15693Type.I_CODE2
 *
 *  Note:
 *
 *      ISO15693_Type = TI HF-I Plus, the range of block = 0 - 63
 *      ISO15693_Type = TI HF-I Pro, the range of block = 0 - 11
 *      ISO15693_Type = I-CODE2, the range of block = 0 - 27
 *****/

private void read_block_iso15693() {
    _WM_RFID rfid = new _WM_RFID();
    //read ISO15693(TI_HF_I_PLUS) , block(10)
    rfid.ReadBlock_ISO15693(context,
        _WM_RFID.ISO15693Type.TI_HF_I_Plus, 10);
}

```

K - 4 Write Block – ISO14443

```

/*****
 *  _WM_RFID API:
 *
 *  WriteBlock_ISO14443(Context context, int blockPage, String writeData)
 *
 *  Parameters :
 *      context      The context of current state of the application/object
 *      blockPage    blockPage(1 - 63)
 *      writeData    writing 32 bytes data to the block of ISO14443
 *
 *  Note:
 *      blockPage = 0, This block saves card ID.
 *      blockPage = 3, 7, 11, 15, 19, ..., 63, Those blocks save AB Key.
 *      Another blocks save user data. ex: 1, 2, 4, 5, 6, 8, ...
 *****/

private void write_block_iso14443() {
    _WM_RFID rfid = new _WM_RFID();
    String data = "0123456789ABCDEF0123456789ABCDEF";
    //write data to block(20)
    //data is "0123456789ABCDEF0123456789ABCDEF"
    rfid.WriteBlock_ISO14443(context, 20, data);
}

```

K - 5 Write Block – ISO15693

```

/*****
*   _WM_RFID API:
*
*   WriteBlock_ISO15693(Context context, ISO15693Type type, int blockPage, String writeData)
*
*   Parameters :
*       context      The context of current state of the application/object
*       type          enum ISO15693Type
*       blockPage     see Note.
*       writeData     writing 16 bytes data to the block of ISO15693
*
*   The type is:
*       public static enum ISO15693Type {
*           TI_HF_I_Plus,
*           TI_HF_I_Pro,
*           I_CODE2
*       }
*
*   Power Saving Mode:
*       1. _WM_RFID.ISO15693Type.TI_HF_I_Plus
*       2. _WM_RFID.ISO15693Type.TI_HF_I_Pro
*       3. _WM_RFID.ISO15693Type.I_CODE2
*
*   Note:
*       ISO15693_Type = TI HF-I Plus, the range of block = 0 - 63
*       ISO15693_Type = TI HF-I Pro, the range of block = 0 - 11
*       ISO15693_Type = I-CODE2, the range of block = 0 - 27
*****/

```

```

private void write_block_iso15693() {
    _WM_RFID rfid = new _WM_RFID();
    String data = "0123456789ABCDEF";
    //write data to ISO15693(TI_HF_I_PLUS), block(20),
    //data is "0123456789ABCDEF"
    rfid.WriteBlock_ISO15693(context,
        _WM_RFID.ISO15693Type.TI_HF_I_Plus, 20, data);
}

```

K - 6 Set A/B Key to RFID reader

```

/*****
*  _WM_RFID API:
*
*  SetABKey(Context context, int keyAddr, String keyData)
*
*  Parameters :
*      context      The context of current state of the application/object
*      keyAddr      Select sector address of RFID reader
*      keyData      writing 12 bytes data to sector address of RFID reader
*
*  Note:
*      keyAddr = 0 - 31
*      keyAddr (0 - 15) is A Key, "0 = A Key1", "1 = A Key2", ..., "15 = A Key16"
*      keyAddr (16 - 31) is B Key, "16 = B Key1", "17 = B Key2", ..., "31 = B Key16"
*
*      keyData is 12 bytes and compose by "0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F"
*****/

private void set_ab_key() {
    _WM_RFID rfid = new _WM_RFID();
    //set A key (address=2), key= "0123456789AB"
    rfid.SetABKey(context, 2, "0123456789AB");
}

```

K - 7 Select A/B Key from RFID reader

```

/*****
 *  _WM_RFID API:
 *
 *  SelectABKey(Context context, int keyAddr)
 *
 *  Parameters :
 *      context      The context of current state of the application/object
 *      keyAddr      Select sector address of RFID reader
 *
 *  Note:
 *      keyAddr = 0 - 31
 *      keyAddr (0 - 15) is A Key, "0 = A Key1", "1 = A Key2", ..., "15 = A Key16"
 *      keyAddr (16 - 31) is B Key, "16 = B Key1", "17 = B Key2", ..., "31 = B Key16"
 *****/

private void select_ab_key() {
    _WM_RFID rfid = new _WM_RFID();
    //select A key (address=2)
    rfid.SelectABKey(context, 2);
}

```

K - 8 Set Trigger Mode

```

/*****
*   _WM_RFID API:
*
*   SetTriggerMode(Context context, TriggerMode mode, TriggerTime time)
*
*   Parameters :
*       context      The context of current state of the application/object
*       mode         enum TriggerMode
*       time         enum TriggerTime
*
*   The type is:
*       public static enum TriggerMode {
*           Auto,
*           Single;
*       }
*
*   The time is:
*       public static enum TriggerTime {
*           Sec_1,
*           Sec_2,
*           Sec_3,
*           Sec_4,
*           Sec_5,
*           Sec_6,
*           Sec_7,
*           Sec_8,
*           Sec_9;
*       }
*   Trigger Mode:
*       1. _WM_RFID.TriggerMode.Auto
*       2. _WM_RFID.TriggerMode.Single
*
*   Trigger Time:
*       1. _WM_RFID.TriggerTime.Sec_1
*       ...
*       ...
*       9. _WM_RFID.TriggerTime.Sec_9
*
*****/

```



```

*   Note:
*
*   If trigger mode is selected "Auto", the trigger time will be ignore.
* *****/

//single trigger
private void set_trigger_mode() {
    _WM_RFID rfid = new _WM_RFID();
    //set trigger to single trigger, trigger time = 2 seconds.
    rfid.SetTriggerMode(mContext,
        _WM_RFID.TriggerMode.Single, _WM_RFID.TriggerTime.Sec_2);
}

//auto trigger
private void set_trigger_mode() {
    _WM_RFID rfid = new _WM_RFID();
    //set trigger to auto trigger, trigger time will be ignored.
    rfid.SetTriggerMode(mContext,
        _WM_RFID.TriggerMode.Auto, _WM_RFID.TriggerTime.Sec_2);
}

```

L. UHF-RFID Settings

Use the following APIs to access the UHF-RFID card.

K - 1 Read UID

K - 2 Read Block – **ISO14443**

Note:

*If WDC responded result is “Y”, it means setting command is successful.
See Appendix A, how to get result from WDC.*

L - 1 Read Tag

```

/*****
*   _WM_RFID API:
*
*   UHF-Read(Context context, int Area)
*
*   Parameters :
*       context      The context of current state of the application/object
*       Area         See note(0 - 3)
*       Length       See note
*
*   Note:
*       Area = 0 - 3
*       Area 0 is Password, Area 1 is EPC, Area 2 is TID, Area 3 is USer Memory
*       Length :
*           When area is Password or EPC or TID the value is 0.
*           When area is User Memory the value is any number, and
*           the maximum value depends on tag type.
*****/

private void uhf_read() {
    _WM_RFID rfid = new _WM_RFID();
    //read TID (Area=2)
    //Length is 0
    rfid.UHF_Read(context, 2, 0);
}

```

L - 2 Write Tag

```

/*****
*   _WM_RFID API:
*
*   UHF-Write(Context context, int Area, String Data)
*
*   Parameters :
*       context      The context of current state of the application/object
*       Area         See note(0 - 2)
*       Data         Write the data to the area of tag, each area has different length
*
*   Note:
*       Area = 0 - 3
*       Area 0 is Password, Area 1 is EPC, Area 2 is USer
*****/

private void uhf_write() {
    _WM_RFID rfid = new _WM_RFID();
    //Write Psssword (Area=0), Data = "12345678"
    rfid.UHF_Write(context, 0, "12345678");
}

```

L - 3 Frequency Specification

```

/*****
*   _WM_RFID API:
*
*   UHF-Write(Context context, int Area, String Data)
*
*   Parameters :
*       context      The context of current state of the application/object
*       Area         See note(0 - 1)
*
*   Note:
*       Area = 0 - 1
*       Area 0 is NA frequency specification, Area 1 is EU frequency specification
*****/

private void uhf_freq_spec() {
    _WM_RFID rfid = new _WM_RFID();
    //Set NA frequency Specification (Area=0)
    rfid.UHF_Freq_Spec(context, 0);
}

```