


	<b>DOTR-900</b>		Date	2013-08-01
			Rev	1.0
			Page	1

# DOTR-900

## ANDROID

**D.O.Tel**




	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	3

## Content


<b>1. INTRODUCTION.....</b>	<b>7</b>
<b>2. DEVELOPMENT TOOL.....</b>	<b>7</b>
<b>3. PROVIDING PACKAGE .....</b>	<b>7</b>
<b>4. ANDRIOD R900 FUNCTION DEFINITION AND EXAMPLE CODE.....</b>	<b>7</b>
<b>4.1 BLUETOOTHACTIVITY.JAVA .....</b>	<b>7</b>
<b>4.1.1 SETONBTEVENTLISTENER.....</b>	<b>7</b>
<b>4.1.2 SCANBLUETOOTHDEVICE.....</b>	<b>8</b>
<b>4.1.3 BYEBLUETOOTHDEVICE .....</b>	<b>9</b>
<b>4.1.4 CONNECTTOBLUETOOTHDEVICE.....</b>	<b>10</b>
<b>4.1.5 SENDCMDOPENINTERFACE1.....</b>	<b>11</b>
<b>4.1.6 SENDCMDINVENTORY .....</b>	<b>11</b>
<b>4.1.7 SENDCMDINVENTORY .....</b>	<b>12</b>
<b>4.1.8 SENDCMDSTOP.....</b>	<b>14</b>
<b>4.1.9 SENDHEARTBEAT.....</b>	<b>15</b>
<b>4.1.10 SENDCMDSELECTMASK.....</b>	<b>15</b>
<b>4.1.11 SENDSETSESSION .....</b>	<b>17</b>
<b>4.1.12 SENDSETQVALUE .....</b>	<b>18</b>
<b>4.1.13 SENDSETINVENTORYTARGET .....</b>	<b>19</b>
<b>4.1.14 SENDINVENTPARAM .....</b>	<b>20</b>
<b>4.1.15 SENDSETSELECTACTION.....</b>	<b>20</b>
<b>4.1.16 SETOPMODE.....</b>	<b>22</b>
<b>4.1.17 SENDREADTAG.....</b>	<b>23</b>
<b>4.1.18 SENDWRITETAG .....</b>	<b>24</b>
<b>4.1.19 CONVERTLOCKINDEX.....</b>	<b>25</b>
<b>4.1.20 SENDLOCKTAG .....</b>	<b>26</b>

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	4


<b>4.1.21 SENDLOCKTAG .....</b>	<b>27</b>
<b>4.1.22 SENDKILLTAG.....</b>	<b>29</b>
<b>4.1.23 SENDGETVERSION .....</b>	<b>30</b>
<b>4.1.24 SENDSETDEFAULTPARAMETER.....</b>	<b>31</b>
<b>4.1.25 SENDGETTINGPARAMETER .....</b>	<b>31</b>
<b>4.1.26 SENDSETTINGTXPOWER .....</b>	<b>32</b>
<b>4.1.27 SENDGETMAXPOWER.....</b>	<b>33</b>
<b>4.1.28 SENDSETTINGTXCYCLE .....</b>	<b>34</b>
<b>4.1.29 SENDCHANGECHANNELSTATE .....</b>	<b>35</b>
<b>4.1.30 SENDSETTINGCOUNTRY .....</b>	<b>36</b>
<b>4.1.31 SENDGETTINGCOUNTRY .....</b>	<b>37</b>
<b>4.1.32 SENDSETLOCKTAGMEMSTATEPERM .....</b>	<b>37</b>
<b>4.1.33 SENDPAUSETX.....</b>	<b>39</b>
<b>4.1.34 SENDSTATUSREPORTING .....</b>	<b>39</b>
<b>4.1.35 SENDINVENTORYREPORTINGFORMAT.....</b>	<b>40</b>
<b>4.1.36 SENDDISLINK.....</b>	<b>41</b>
<b>4.1.37 SENDUPLOADINGTAGDATA .....</b>	<b>42</b>
<b>4.1.38 SENDCLEARINGTAGDATA .....</b>	<b>43</b>
<b>4.1.39 SENDALERTREADERSTATUS.....</b>	<b>44</b>
<b>4.1.40 SENDGETTINGSTATUSWORD.....</b>	<b>45</b>
<b>4.1.41 SENDSETTINGBUZZERVOLUME.....</b>	<b>46</b>
<b>4.1.42 SENDBEEP .....</b>	<b>47</b>
<b>4.1.43 SENDSETTINGAUTOPOWEROFFDELAY.....</b>	<b>48</b>
<b>4.1.44 SENDGETTINGBATTERYLEVEL .....</b>	<b>48</b>
<b>4.1.45 SENDREPORTINGBATTERYSTATE .....</b>	<b>49</b>
<b>4.1.46 SENDTURNINGREADEROFF .....</b>	<b>50</b>

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	5

<b>4.2 ONBTEVENTLISTENER.JAVA.....</b>	<b>51</b>
4.2.1 ONBTFOUNDNEWDEVICE.....	51
4.2.2 ONBTSCANCOMPLETED.....	52
4.2.3 ONBTCONNECTED.....	52
4.2.4 ONBTDISCONNECTED.....	52
4.2.5 ONBTCONNECTFAIL.....	53
4.2.6 ONBTDATASENT .....	53
4.2.7 ONBTDATATRANSEXCEPTION.....	54
4.2.8 ONNOTIFYBTDATARECV .....	54
<b>4.3 R900MANAGER.JAVA .....</b>	<b>55</b>
4.3.1 ISBLUETOOTHENABLED.....	55
4.3.2 ENABLEBLUETOOTH .....	55
4.3.3 QUERYPAIREDDEVICES .....	56
4.3.4 STARTDISCOVERY .....	57
4.3.5 STOPDISCOVERY .....	58
4.3.6 GETBLUETOOTHDEVICE .....	58
4.3.7 SENDDATA .....	59
4.3.8 ISTRYINGCONNECT.....	60
<b>4.4 R900PROTOCOL.JAVA. ....</b>	<b>61</b>
4.4.1 MAKEPROTOCOL .....	61
4.4.2 MAKEPROTOCOL .....	62
4.4.3 MAKEPROTOCOL .....	63
4.4.4 MAKEPROTOCOL .....	64
4.4.6 MAKEPROTOCOL .....	65
4.4.7 MAKEPROTOCOL .....	66
4.4.8 STRING2BYTES .....	67

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	6

<b>4.5 R900RECVPACKETPARSER.JAVA .....</b>	<b>68</b>
<b>4.5.1 PUSHPACKET .....</b>	<b>68</b>
<b>4.5.2 POPPACKET.....</b>	<b>69</b>
<b>4.5.3 POPPACKET.....</b>	<b>70</b>
<b>5. HOW TO INSTALL A DEMO.....</b>	<b>72</b>
<b>6. HOW TO USE A DEMO.....</b>	<b>73</b>

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	7

## 1. INTRODUCTION

This document describes the API function and usage of DOTR-900.

The DOTR-900 is a UHF reader for 900MHz and it can connect to various devices by using Bluetooth.

This document explains how to connect between DOTR-900 and Android device.

A user can access easily by API function and example source.

Also, this document includes how to use DEMO application.

## 2. Development Tool

### Development Platform

Add the JAVA platform for developing DOTR-900.

Download to install applications by below sequence.

- Download a “android\_sdk\_r09\_windows.zip” from [www.Developer.android.com](http://www.Developer.android.com)

- **Eclipse : Indigo version**

[Download Eclipse](#)

- **JAVA : jdk1.6.0\_23**

[Download JAVA](#)

- **ADT12.0.0**

Click a SDK in [www.Developer.android.com](http://www.Developer.android.com) and find to download ADT12.0.0

[Download ADT](#)

- **Android SDK and AVD Manager**

If “SDK Platform Android 2.2 API8, revision 2” is not installed, use “SDK Platform Android 2.2, revision 2”

## 3. Providing Package

The provided package when developing is provided with below table.

Device	Provided Document
R900	R900 Manual

## 4. Definition of Android function and example for R900

### 4.1 BluetoothActivity.java

When RFID activates, it works based on Bluetooth.

#### 4.1.1 setOnBtEventListener

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	8

This function is to register an event from application.

---

**public void setOnBtEventListener( OnBtEventListener listener )**

---

#### Parameters

OnBtEventListener listener : The created object currently

#### Return Values

None

#### Remarks

This function is provided for transmitting/receiving a data, inquiry and adding a Bluetooth device

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
private OnBtEventListener mBtEventListener;
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

mBtEventListener = listener;

public void setOnBtEventListener( OnBtEventListener listener )
{
    mBtEventListener = listener;
    if( mR900Manager != null )
        mR900Manager.setOnBtEventListener(listener);
}
```

---

#### 4.1.2 scanBluetoothDevice

This function is for inquiry around Bluetooth devices.

---


**public void scanBluetoothDevice()**

---

#### Parameters

None



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	9

#### Return Values

None

#### Remarks

An event is registered and used by SetOnBtEventListener() before using this function.

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void scanBluetoothDevice()
{
    if( mR900Manager != null )
        mR900Manager.startDiscovery();
}
```

#### 4.1.3 byeBluetoothDevice

This function is used to release the connection from a DOTR-900.

```
public void byeBluetoothDevice()
```

#### Parameters

None

#### Return Values

None


#### Remarks

None

#### Example Code

##### JAVA

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	10

### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void byeBluetoothDevice()
{
    if( mR900Manager != null )
        mR900Manager.sendData(R900Protocol.BYE);
}
```

#### 4.1.4 connectToBluetoothDevice

This function is to connect to Bluetooth device.

```
public void connectToBluetoothDevice( String address, UUID uuid )
```

#### Parameters

address

MAC address of Bluetooth device desired to connect

uuid

Bluetooth device's ID to connect

#### Return Values

None

#### Remarks

In case of first connection, the pairing operation is progressed.


#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void connectToBluetoothDevice( String address, UUID uuid )
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	11

```
{
  if( mR900Manager != null && address != null )
    mR900Manager.connectToBluetoothDevice(address, uuid);
}
```

#### 4.1.5 sendCmdOpenInterface1

This function is to open a DOTR-900.

```
public void sendCmdOpenInterface1()
```

##### Parameters

None

##### Return Values

None

##### Remarks

This function must be used with connection state by connectToBluetoothDevice().

##### Example Code

###### JAVA

###### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendCmdOpenInterface1()
{
  if( mR900Manager != null )
    mR900Manager.sendData(R900Protocol.OPEN_INTERFACE_1);
}
```

#### 4.1.6 sendCmdInventory

This function is to inventory Tags.

```
public void sendCmdInventory()
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	12

#### Parameters

None

#### Return Values

None

#### Remarks

Refer 4.1.7

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected boolean mSingleTag;
protected boolean mUseMask;
protected int mTimeout;
protected boolean mQuerySelected;
```

```
protected String mLastCmd;
```

```
import com.dotel.rfid.R900Status;
```

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void sendCmdInventory()
{
    sendCmdInventory( mSingleTag ? 1 : 0, mUseMask ? ( mQuerySelected ? 3 : 2 ) : 0, mTimeout );
}
```

#### 4.1.7 sendCmdInventory


This function is to inventory Tags.

```
public void sendCmdInventory( int f_s, int f_m, int to )
```

#### Parameters

f\_s

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	13

Set to recognize only one Tag or more.

0 : Recognize Tags continuously

1 : Recognize only one Tag.

f\_m

Set to select a mask

0 or 1 : no select a mask

2 : Recognize Tags that doesn't select a mask only

3 : Recognize Tags that selects a mask only

to

This parameter is timeout value for inventory duration and the unit is msec.

In case of 0, it means sustained operation.

#### Return Values

None

#### Remarks

Can set Tag through sendCmdSelectMask() to recognize desired.

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected boolean mSingleTag;
protected boolean mUseMask;
protected int mTimeout;
protected boolean mQuerySelected;
```

```
protected String mLastCmd;
```

```
import com.dotel.rfid.R900Status;
```

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void sendCmdInventory( int f_s, int f_m, int to )
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	14

```

{
  if( mR900Manager != null )
  {
    R900Status.setOperationMode(1);
    mLastCmd = R900Protocol.CMD_INVENT;
    mR900Manager.sendData(R900Protocol.makeProtocol(mLastCmd, new int[] { f_s, f_m, to }));
  }
}

```

#### 4.1.8 sendCmdStop

This function is to stop current operation.

**public void sendCmdStop()**

##### Parameters

None

##### Return Values

None

##### Remarks

None

##### Example Code

###### JAVA

###### BluetoothActivity.java

```

import com.dotel.rfid.R900Status;

protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendCmdStop()
{
  if( mR900Manager != null )
  {
    R900Status.setOperationMode(0);
    mLastCmd = R900Protocol.CMD_STOP;
    mR900Manager.sendData(R900Protocol.makeProtocol(mLastCmd, (int[])null));
  }
}

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	15

}

#### 4.1.9 sendHeartBeat

This function is to set for checking connection status with Host

**public void sendHeartBeat( int value )**

##### Parameters

value

This parameter is time value for checking connection status and the unit is msec.

In case of 0, don't check connection status.

##### Return Values

None

##### Remarks

None

##### Example Code

###### JAVA

###### BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendHeartBeat( int value )
```

```
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol(R900Protocol.CMD_HEART_BEAT,
            new int[]{ value }));
    }
}
```

#### 4.1.10 sendCmdSelectMask

This function is to set for recognizing the selected Tag.

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	16

---

**public void sendCmdSelectMask( int n, int bits, int mem, int b\_offset, String pattern, int target, int action )**

---

#### Parameters

n

Set a index for mask table with 0~7 range.

bits

Set bit of mask pattern. In case of 0, release a mask.

mem

Set a Tag's memory area.

0 : Reserved area

1 : EPC area

2 : TID area

3 : User area

b\_offset

Set a start bit of Tag's memory when apply a mask pattern.

In case of EPC, start of memory is 16 bit.

pattern

Set a Hex string of applicable mask pattern

target

Set a Flag value to apply for action. The value for recognizing is 4.

action

Set a flag of the selected TAG. Refer Appendix I for Flag.

#### Return Values

None

#### Remarks

None

#### Example Code

**JAVA**

**BluetoothActivity.java**

```
protected R900Manager mR900Manager;
```

---



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	17

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendCmdSelectMask( int n, int bits, int mem, int b_offset, String pattern, int target, int action )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_SEL_MASK,
            new int[]{ n, bits, mem, b_offset}, pattern, new int[]{ target, action } ));
    }
}
```

#### 4.1.11 sendSetSession

This function is to set session for inventory.

```
public void sendSetSession( int session )
```

#### Parameters

session

Indicate a session value when query a TAG. The range of session is from 0 to 3 and the basic value is 0.

0 : The recognized TAG Flag changes to previous value within 2 seconds. (Session 0)

1 : The recognized TAG Flag changes to previous value within 2~5 seconds. (Session 1)

2 : The recognized TAG Flag changes to previous value 5 seconds later. (Session 2)

3 : The recognized TAG Flag doesn't changes to previous value. (Session 3)

#### Return Values

None

#### Remarks

None


#### Example Code

**JAVA**

**BluetoothActivity.java**

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	18

```

public void sendSetSession( int session )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_INVENT_PARAM,
        new int[]{ session, R900Protocol.SKIP_PARAM, R900Protocol.SKIP_PARAM } ) );
    }
}

```

#### 4.1.12 sendSetQValue

This function is to set Q value for inventory.

```

public void sendSetQValue( int q )

```

#### Parameters

q

Set a Q value when query a TAG. The range of Q value is from 0 to 15 and the basic value is 5.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA


##### BluetoothActivity.java

```

protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendSetQValue( int q )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_INVENT_PARAM,
        new int[]{ R900Protocol.SKIP_PARAM, q, R900Protocol.SKIP_PARAM } ) );
    }
}

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	19

#### 4.1.13 sendSetInventoryTarget

This function is to indicate status of flag for inventory.

---

```
public void sendSetInventoryTarget( int m_ab )
```

---

#### Parameters

m\_ab

Indicate status of inventory flag when query a TAG. The range is from 0 to 2 and the basic value is 2.

0 : Inventory flag A

1 : Inventory flag B

2 : Inventory flag A or B

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendSetInventoryTarget( int m_ab )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_INVENT_PARAM,
            new int[]{ R900Protocol.SKIP_PARAM, R900Protocol.SKIP_PARAM, m_ab } ) );
    }
}
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	20

#### 4.1.14 sendInventParam

This function is to set parameters for inventory.

---

```
public void sendInventParam( int session, int q, int m_ab )
```

---

#### Parameters

session

Indicate a session when query a TAG. The range is from 0 to 3 and the basic value is 0.

q

Indicate a Q value when query a TAG. The range is from 0 to 15 and the basic value is 5.

m\_ab

Indicate a TAG inventory flag when query a TAG. The range is from 0 to 2 and the basic value is 2.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```


```
public void sendInventParam( int session, int q, int m_ab )
{
if( mR900Manager != null )
{
    mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_INVENT_PARAM,
    new int[]{ session, q, m_ab } ) );
}
}
```

---

#### 4.1.15 sendSetSelectAction

This function is to set for selecting a mask when inventory.

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	21

---

**public void sendSetSelectAction( int bits, int mem, int b\_offset, String pattern, int action )**

---

#### Parameters

bits

Set a bit of mask pattern. In case of 0, release a mask.

mem

Set a TAG's memory area.

0 : Reserved area

1 : EPC area

2 : TID area

3 : User area

b\_offset

Set a start bit of Tag's memory when apply a mask pattern.

In case of EPC, start of memory is 16 bit.

pattern

Set a Hex string of applicable mask pattern.

action

Set a flag of the selected TAG. Refer Appendix I for Flag.

#### Return Values

None

#### Remarks

None

#### Example Code


##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendSetSelectAction( int bits, int mem, int b_offset, String pattern, int action )
{
    if( mR900Manager != null )
    {
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	22

```

mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_SEL_MASK,
    new int[]{ 0, bits, mem, b_offset}, pattern, new int[]{ R900Protocol.SKIP_PARAM, action } ));
    }
}

```

#### 4.1.16 setOpMode

This function is to set operation parameters for inventory.

```
public void setOpMode( boolean singleTag, boolean useMask, int timeout, boolean querySelected )
```

#### Parameters

singleTag

Set a number for recognizing a TAG.

True ; Recognize TAGs continuously

False : Recognize only one TAG

useMask

Set a mask operation.

True : Recognize only the selected TAGs

False : Recognize TAGs randomly

timeout

Set a timeout value for recognizing duration and the unit is msec.

In case of 0, recognize TAGs continuously.

querySelected

Set a selection for query.

#### Return Values

None

#### Remarks


None

#### Example Code

JAVA

BluetoothActivity.java

```
protected boolean mSingleTag;
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	23

```
protected boolean mUseMask;
protected int mTimeout;
protected boolean mQuerySelected;
```

```
public void setOpMode( boolean singleTag, boolean useMask, int timeout, boolean querySelected )
{
    mSingleTag = singleTag;
    mUseMask = useMask;
    mTimeout = timeout;
    mQuerySelected = querySelected;
}
```

#### 4.1.17 sendReadTag

This function is to print a data from the read TAG.

```
public void sendReadTag( int w_count, int mem, int w_offset, String ACS_PWD )
```

#### Parameters

w\_count

Set a data length from the read TAG. The unit is word(16bit) and the basic value is 1.

mem

Indicate a memory location from the read TAG.

0 : RESERVED area

1 : EPC area

2 : TID area

3 : USER area

w\_offset

Set a start location of memory from the read TAG. The unit is word(16bit).

ACS\_PWD

Set the stored access password from a TAG. The basic value is 0.

#### Return Values

None

#### Remarks

None

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	24

## Example Code

### JAVA

#### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendReadTag( int w_count, int mem, int w_offset, String ACS_PWD )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_READ_TAG_MEM,
        new int[]{ w_count, mem, w_offset }, ACS_PWD,
        new int[]{ mSingleTag ? 1 : 0, mUseMask ? ( mQuerySelected ? 3 : 2 ) : 0, mTimeout } ) );
    }
}
```

#### 4.1.18 sendWriteTag

This function is to record a data to a TAG.

```
public void sendWriteTag( int w_count, int mem, int w_offset, String ACS_PWD, String  
wordPattern )
```

#### Parameters

w\_count

Set a data length to record to a TAG. The unit is word(16bit).

mem

태그에 기록하기 위한 메모리 위치를 지정합니다.

Indicate a memory location to record to a TAG.

0 : RESERVED area

1 : EPC area

2 : TID area

3 : USER area

w\_offset

Indicate a start location of memory to record to a TAG. The unit is word(16bit).

ACS\_PWD

Indicate the stored access password from a TAG.



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	25

wordPattern

Indicate a data to record to a TAG. The data is Hex String.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected boolean mSingleTag;
protected boolean mUseMask;
protected int mTimeout;
protected boolean mQuerySelected;
```

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void sendWriteTag( int w_count, int mem, int w_offset, String ACS_PWD, String wordPattern )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_WRITE_TAG_MEM,
        new int[]{ w_count, mem, w_offset }, new String[]{ wordPattern, ACS_PWD },
        new int[]{ mSingleTag ? 1 : 0, mUseMask ? ( mQuerySelected ? 3 : 2 ) : 0, mTimeout } ) );
    }
}
```

#### 4.1.19 convertLockIndex

This function is to convert a lock index.

```
private int convertLockIndex( boolean enable, boolean index )
```

#### Parameters

enable

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	26

Indicate usage for a lock index.

0 : Deactivate a lock index

1 : Activate a lock index

index

Indicate a location of lock index.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
private int convertLockIndex( boolean enable, boolean index )
{
    return enable ? ( index ? 1 : 0 ) : -1;
}
```

#### 4.1.20 sendLockTag

This function is to set a locking operation for specific memory.

```
public void sendLockTag( LockPattern lockPattern, String ACS_PWD )
```

#### Parameters

lockPattern

If this parameter is 0, release a locking operation. In case of 1, set a locking operation.

kill\_pwd : Indicate a locking operation of Kill password


acs\_pwd : Indicate a locking operation of Access password

epc : Indicate a locking operation of EPC memory

tid : Indicate a locking operation of TID memory

user : Indicate a locking operation of user memory

ACS\_PWD

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	27

Indicate the stored access password from a TAG.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java


```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendLockTag( LockPattern lockPattern, String ACS_PWD )
{
    if( mR900Manager != null )
    {
        final int user = convertLockIndex( lockPattern.enableUser,
        lockPattern.indexUser );//( ( lockPattern.indexUser == false ) ? ( R900Protocol.SKIP_PARAM ) :
        ( lockPattern.enableUser ? 1 : 0 ) );
        final int tid = convertLockIndex( lockPattern.enableTid, lockPattern.indexTid );//( ( lockPattern.indexTid
        == false ) ? ( R900Protocol.SKIP_PARAM ) : ( lockPattern.enableTid ? 1 : 0 ) );
        final int epc = convertLockIndex( lockPattern.enableUii, lockPattern.indexUii );//( ( lockPattern.indexUii
        == false ) ? ( R900Protocol.SKIP_PARAM ) : ( lockPattern.enableUii ? 1 : 0 ) );
        final int acs_pwd = convertLockIndex( lockPattern.enableAcsPwd,
        lockPattern.indexAcsPwd );//( ( lockPattern.indexAcsPwd == false ) ? ( R900Protocol.SKIP_PARAM ) :
        ( lockPattern.enableAcsPwd ? 1 : 0 ) );
        final int kill_pwd = convertLockIndex( lockPattern.enableKillPwd,
        lockPattern.indexKillPwd );//( ( lockPattern.indexKillPwd == false ) ? ( R900Protocol.SKIP_PARAM ) :
        ( lockPattern.enableKillPwd ? 1 : 0 ) );

        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_LOCK_TAG_MEM,
        new int[]{ user, tid, epc, acs_pwd, kill_pwd }, ACS_PWD,
        new int[]{ mSingleTag ? 1 : 0, mUseMask ? ( mQuerySelected ? 3 : 2 ) : 0, mTimeout } ) );
    }
}
```

#### 4.1.21 sendLockTag

This function is to set a locking operation for specific memory.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	28

---

**public void sendLockTag( int lockMask, int lockEnable, String ACS\_PWD )**

---

#### Parameters

lockMask

Indicate a mask value to set a locking operation for specific memory of TAG.

lockEnable

Indicate a mask field value to set a locking operation for specific memory of TAG.

ACS\_PWD

Indicate the stored access password from a TAG.

#### Return Values

None

#### Remarks

None


#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendLockTag( int lockMask, int lockEnable, String ACS_PWD )
{
    if( mR900Manager != null )
    {
        int bitFlag;
        boolean mask;
        boolean enable;
        //--
        bitFlag = 0x200;//0x02;
        mask = ( lockMask & bitFlag ) == bitFlag;
        enable = ( lockEnable & bitFlag ) == bitFlag;
        final int user = ( ( mask == false ) ? ( R900Protocol.SKIP_PARAM ) : ( enable ? 1 : 0 ) );
        //--
        bitFlag = 0x80;//0x08;
        mask = ( lockMask & bitFlag ) == bitFlag;
        enable = ( lockEnable & bitFlag ) == bitFlag;
        final int tid = ( ( mask == false ) ? ( R900Protocol.SKIP_PARAM ) : ( enable ? 1 : 0 ) );
        //--
    }
}
```

	<h1>DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	29

```

bitFlag = 0x20;
mask = ( lockMask & bitFlag ) == bitFlag;
enable = ( lockEnable & bitFlag ) == bitFlag;
final int epc = ( ( mask == false ) ? ( R900Protocol.SKIP_PARAM ) : ( enable ? 1 : 0 ) );
//---
bitFlag = 0x08;//0x80;
mask = ( lockMask & bitFlag ) == bitFlag;
enable = ( lockEnable & bitFlag ) == bitFlag;
    final int acs_pwd = ( ( mask == false ) ? ( R900Protocol.SKIP_PARAM ) : ( enable ? 1 : 0 ) );
//---
bitFlag = 0x02;//0x200;
mask = ( lockMask & bitFlag ) == bitFlag;
enable = ( lockEnable & bitFlag ) == bitFlag;
final int kill_pwd = ( ( mask == false ) ? ( R900Protocol.SKIP_PARAM ) : ( enable ? 1 : 0 ) );
//---
    mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_LOCK_TAG_MEM,
new int[] { user, tid, epc, acs_pwd, kill_pwd }, ACS_PWD,
    new int[] { mSingleTag ? 1 : 0, mUseMask ? ( mQuerySelected ? 3 : 2 ) : 0, mTimeout } ) );
}
}

```

#### 4.1.22 sendKillTag

This function is to disable using a TAG

```
public void sendKillTag( String killPwd )
```

#### Parameters

killPwd

Indicate the stored kill password from a TAG. In case of 0, it doesn't work.

#### Return Values

None

#### Remarks

None

#### Example Code

JAVA

BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	30

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendKillTag( String killPwd )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_KILL_TAG,
            killPwd, new int[]{ mSingleTag ? 1 : 0, mUseMask ? ( mQuerySelected ? 3 : 2 ) : 0, mTimeout } ) );
    }
}
```

#### 4.1.23 sendGetVersion

This function is to verify a F/W version of DOTR-900.

```
public void sendGetVersion()
```

##### Parameters

None

##### Return Values

None

##### Remarks

An event is registered and used by SetOnBtEventListener() before using this function.

##### Example Code

###### JAVA

###### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendGetVersion()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_GET_VERSION ) );
    }
}
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	31

#### 4.1.24 sendSetDefaultParameter

This function is to recover the changed parameters after reset of DOTR-900.

---

**public void sendSetDefaultParameter()**

---

##### Parameters

None

##### Return Values

None

##### Remarks

None

##### Example Code

**JAVA**

**BluetoothActivity.java**

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendSetDefaultParameter()
```

```
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_SET_DEF_PARAM ) );
    }
}
```

---

#### 4.1.25 sendGettingParameter

This function is to verify the applied parameters for inventory.


---

**public void sendGettingParameter( String cmd, String p )**

---

##### Parameters

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	32

None

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendGettingParameter( String cmd, String p )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_GET_PARAM, new
        String[]{ cmd, p } ) );
    }
}
```

#### 4.1.26 sendSettingTxPower

This function is to set a output power level.

```
public void sendSettingTxPower( int a )
```

#### Parameters

a

This parameter is decrease of output power and is set by 0 or minus value. The unit is dB.


#### Return Values

None

#### Remarks

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	33

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendSettingTxPower( int a )
{
    if( mR900Manager != null )
    {

        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_SET_TX_POWER, new
        int[] { a } ) );
    }
}
```

#### 4.1.27 sendGetMaxPower

This function is to verify the maximum output power level of DOTR-900.

```
public void sendGetMaxPower()
```

#### Parameters

None

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	34

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void sendGetMaxPower()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_GET_MAX_POWER ) );
    }
}
```

#### 4.1.28 sendSettingTxCycle

This function is used to indicate delivery duration on a channel.

```
public void sendSettingTxCycle( int on, int off)
```

#### Parameters

on

Indicate the maximum delivery duration. The unit is msec.

off

Indicate the maximum off time. The unit is msec.

#### Return Values

None

#### Remarks

This is different according to rule of country.

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

```
public void sendSettingTxCycle( int on, int off)
{
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	35

```

if( mR900Manager != null )
{
    mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_SET_TX_CYCLE, new int[] { on,
    off } ) );
}
}

```

#### 4.1.29 sendChangeChannelState

This function is used to set usage of a channel.

```
public void sendChangeChannelState( int n, int f_e )
```

##### Parameters

n

Set a number of channel.

f\_e

채널 사용 여부를 설정합니다.

Set usage of a channel.

0 : Disable usage of a channel

1 : Enable usage of a channel

##### Return Values

None

##### Remarks

None

##### Example Code

###### JAVA


###### BluetoothActivity.java

```

protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendChangeChannelState( int n, int f_e )
{
    if( mR900Manager != null )

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	36

```

{
    mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_CHANGE_CH_STATE, new
    int[] { n, f_e } ) );
}
}

```

#### 4.1.30 sendSettingCountry

This function is to set the region for module operation.

```
public void sendSettingCountry( int code )
```

#### Parameters

code

Set a country code.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendSettingCountry( int code )
```

```

{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_CHANGE_CH_STATE, new
        int[] { code } ) );
    }
}

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	37

#### 4.1.31 sendGettingCountry

This function is to verify the region for module operation.

---

**public void sendGettingCountry()**

---

##### Parameters

None

##### Return Values

None

##### Remarks

None

##### Example Code

**JAVA**

**BluetoothActivity.java**

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendGettingCountry()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_GET_COUNTRY_CAP ) );
    }
}
```

---

#### 4.1.32 sendSetLockTagMemStatePerm

This function is to provide a TAG's specific memory to change permanently.

---


**public void sendSetLockTagMemStatePerm( int mem\_id, int f\_l, String ACS\_PWD )**

---

##### Parameters

mem\_id

Indicate a memory.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	38

0 : USER area

1 : TID area

2 : EPC area

3 : Access password area

4 : Kill password area

f\_1

Set a memory to change permanently

0 : Change

1 : Change permanently

ACS\_PWD

Indicate a TAG's access password.

#### Return Values

None

#### Remarks

Once, memory is changed permanently, can't change any more.

The changed memory permanently doesn't accept writing and have to use an access password to write.

#### Example Code

##### JAVA


##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendSetLockTagMemStatePerm( int mem_id, int f_1, String ACS_PWD )
```

```
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_SET_LOCK_TAG_MEM,
            new int[]{ mem_id, f_1 }, ACS_PWD,
            new int[]{ mSingleTag ? 1 : 0, mUseMask ? ( mQuerySelected ? 3 : 2 ) : 0, mTimeout } ) );
    }
}
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	39

#### 4.1.33 sendPauseTx

This function is to stop a data transmission of DOTR-900.

---

**public void sendPauseTx()**

---

##### Parameters

None

##### Return Values

None

##### Remarks

None

##### Example Code

**JAVA**

**BluetoothActivity.java**

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendPauseTx()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_PAUSE_TX ) );
    }
}
```

---

#### 4.1.34 sendStatusReporting

This function is to report status of DOTR-900.

---


**public void sendStatusReporting( int f\_link )**

---

##### Parameters

f\_link

Set usage of the report operation for DOTR-900.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	40

0 : Enable  
1 : Disable

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendStatusReporting( int f_link )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_STATUS_REPORT,
        new int[] { f_link } ) );
    }
}
```

#### 4.1.35 sendInventoryReportingFormat

This function is to indicate a inventory report format.

```
public void sendInventoryReportingFormat( int f_time, int f_rssi )
```

#### Parameters

f\_time

Print a time for inventory.


f\_rssi

Print a RSSI value for inventory.

#### Return Values

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	41

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendInventoryReportingFormat( int f_time, int f_rssi )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_INVENT_REPORT_FORMAT,
            new int[]{ f_time, f_rssi } ) );
    }
}
```

#### 4.1.36 sendDislink

This function is to release the open status of DOTR-900.

```
public void sendDislink()
```

#### Parameters

None

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	42

## BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendDislink()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_DISLINK ) );
    }
}
```

### 4.1.37 sendUploadingTagData

This function is to upload the TAG information from local mode of DOTR-900.

```
public void sendUploadingTagData( int index, int count )
```

#### Parameters

index

Indicate the data list. The basic value is 0.

count

Indicate the data number.

#### Return Values

None

#### Remarks

None

#### Example Code

**JAVA**

**BluetoothActivity.java**

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	43

```

public void sendUploadingTagData( int index, int count )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_UPLOAD_TAG_DATA,
            new int[]{ index, count } ) );
    }
}

```

#### 4.1.38 sendClearingTagData

This function is to clear the stored TAG information in DOTR-900.

```
public void sendClearingTagData()
```

##### Parameters

None

##### Return Values

None

##### Remarks

None

##### Example Code

###### JAVA


###### BluetoothActivity.java

```

protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendClearingTagData()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_CLEAR_TAG_DATA ) );
    }
}

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	44

#### 4.1.39 sendAlertReaderStatus

This function is to set usage of the report when DOTR-900 is changed.

---

```
public void sendAlertReaderStatus( int f_link, int f_trigger, int f_lowbat, int f_autooff, int f_pwr )
```

---

#### Parameters

f\_link

Indicate usage of the report according status of link

0 : Disable

1 : Enable

f\_trigger

Indicate usage of the report according to trigger button

0 : Disable

1 : Enable

f\_lowbat

Indicate usage of the report according to status of battery

0 : Disable

1 : Enable

f\_autooff

Indicate usage of the report according to status of DOTR-900 auto-off

0 : Disable

1 : Enable

f\_pwr

Indicate usage of the report according to terminate DOTR-900.

0 : Disable

1 : Enable

#### Return Values

None

#### Remarks

None

#### Example Code

**JAVA**

---

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	45

## BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendAlertReaderStatus( int f_link, int f_trigger, int f_lowbat, int f_autooff, int f_pwr )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_ALERT_READER_STATUS,
            new int[] { f_link, f_trigger, f_lowbat, f_autooff, f_pwr } ) );
    }
}
```

---

### 4.1.40 sendGettingStatusWord

This function is to print status value of DOTR-900.

---

**public void sendGettingStatusWord()**

---

#### Parameters

None

#### Return Values

None

#### Remarks

None


#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendGettingStatusWord()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_GET_STATUS_WORD ) );
    }
}
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	46

```

}
}

```

#### 4.1.41 sendSettingBuzzerVolume

This function is to change a buzzer volume value of DOTR-900.

```
public void sendSettingBuzzerVolume( int volume, int f_nv )
```

#### Parameters

volume

Indicate a volume value.

0 : Mute

1 : middle

2 : high

f\_nv

Indicate usage of an EEPROM.

0 : Recover to last status when turn-off

1 : Retain current status when turn-off

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA


##### BluetoothActivity.java

```

protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendSettingBuzzerVolume( int volume, int f_nv )
{
    if( mR900Manager != null )
    {

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	47

```

mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_SET_BUZZER_VOL,
new int[] { volume, f_nv } ) );
}
}

```

#### 4.1.42 sendBeep

This function is to make a beep sound.

```
public void sendBeep( int f_on )
```

#### Parameters

f\_on

Indicate usage of a beep sound.

0 : Disable

1 : Enable

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA


##### BluetoothActivity.java

```

protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendBeep( int f_on )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_BEEP,
new int[] { f_on } ) );
    }
}

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	48

#### 4.1.43 sendSettingAutoPowerOffDelay

This function is to change a time value for auto termination of DOTR-900.

---

```
public void sendSettingAutoPowerOffDelay( int delay, int f_nv )
```

---

##### Parameters

delay

Indicate a time value for termination. The unit is msec.

f\_nv

Indicate usage of an EEPROM.

0 : Recover to last status when turn-off

1 : Retain current status when turn-off

##### Return Values

None

##### Remarks

None

##### Example Code

###### JAVA

###### BluetoothActivity.java

```
protected R900Manager mR900Manager;
```

```
mR900Manager = new R900Manager( mHandler );
```

```
public void sendSettingAutoPowerOffDelay( int delay, int f_nv )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol(
            R900Protocol.CMD_SET_AUTO_POWER_OFF_DELAY, new int[]{ delay, f_nv } ) );
    }
}
```

---

#### 4.1.44 sendGettingBatteryLevel

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.



	<h1>DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	49

This function is to check life of a battery for the DOTR-900.

---

**public void sendGettingBatteryLevel( int f\_ext )**

---

#### Parameters

f\_ext

Indicate the test information. The basic value is 0.

#### Return Values

None

#### Remarks

This function is for engineering.

#### Example Code

**JAVA**

**BluetoothActivity.java**

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendGettingBatteryLevel( int f_ext )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_GET_BATT_LEVEL,
            new int[]{ f_ext } ) );
    }
}
```

---

#### 4.1.45 sendReportingBatteryState

This function is to report status of a battery for the DOTR-900.

---

**public void sendReportingBatteryState( int f\_report )**


---

#### Parameters

f\_report

Indicate usage of the report operation.

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	50

0 : Disable

1 : Enable

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendReportingBatteryState( int f_report )
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_REPORT_BATT_STATE,
            new int[]{ f_report } ) );
    }
}
```

#### 4.1.46 sendTurningReaderOff

This function is to terminate the DOTR-900.

```
public void sendTurningReaderOff()
```

#### Parameters


None

#### Return Values

None

#### Remarks

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	51

None

## Example Code

### JAVA

#### BluetoothActivity.java

```
protected R900Manager mR900Manager;
mR900Manager = new R900Manager( mHandler );

public void sendTurningReaderOff()
{
    if( mR900Manager != null )
    {
        mR900Manager.sendData(R900Protocol.makeProtocol( R900Protocol.CMD_TURN_READER_OFF ) );
    }
}
```

## 4.2 OnBtEventListener.java

This code is for event of Bluetooth connection.

### 4.2.1 onBtFoundNewDevice

This function is used to register new device when Bluetooth inquiry.

```
void onBtFoundNewDevice( BluetoothDevice device )
```

#### Parameters

device


Indicate the information of Bluetooth device.

#### Return Values

None

#### Remarks

None

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	52

#### 4.2.2 onBtScanCompleted

This function is to deliver an event for scan completion of a Bluetooth device.

---

**void onBtScanCompleted()**

---

##### Parameters

None

##### Return Values

None

##### Remarks

None

---

#### 4.2.3 onBtConnected

This function is for Bluetooth connection.

---

**void onBtConnected( BluetoothDevice device )**

---

##### Parameters

device

Indicate the information of Bluetooth device.

##### Return Values

None

##### Remarks

None

---

#### 4.2.4 onBtDisconnected

This function is to release Bluetooth connection.

---

**void onBtDisconnected( BluetoothDevice device )**


---

##### Parameters

---

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

---

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	53

device

Indicate the information of Bluetooth device.

#### Return Values

None

#### Remarks

None

#### 4.2.5 onBtConnectFail

This function is to deliver an event for connection failure of Bluetooth.

**void onBtConnectFail( BluetoothDevice device, String msg )**

#### Parameters

device

Indicate the information of Bluetooth device.

msg

Deliver a message.

#### Return Values

None

#### Remarks

None

#### 4.2.6 onBtDataSent


This function is to deliver an event for data transmission to Bluetooth device.

**void onBtDataSent( byte[] data )**

#### Parameters

data

Indicate a data from Bluetooth device.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	54

#### Return Values

None

#### Remarks

None

#### 4.2.7 onBtDataTransException

This function is to deliver an event for exception data of Bluetooth.

**void onBtDataTransException( BluetoothDevice device, String msg )**

#### Parameters

device

Indicate the information of Bluetooth.

msg

Indicate a message for exception.

#### Return Values

None

#### Remarks

None

#### 4.2.8 onNotifyBtDataRecv

This function is to notify an event for the received data from Bluetooth device.


**void onNotifyBtDataRecv()**

#### Parameters

None

#### Return Values

None

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	55

#### Remarks

None

### 4.3 R900Manager.java

This code is to control connection with the DOTR-900.

#### 4.3.1 isBluetoothEnabled

This function is to check whether Bluetooth is enabled.

---

**public boolean isBluetoothEnabled()**

---

#### Parameters

None

#### Return Values

None

#### Remarks

None

#### Example Code

**JAVA**

**R900Manager.java**

```
private BluetoothAdapter mBluetoothAdapter;
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

public boolean isBluetoothEnabled()
{
    if( mBluetoothAdapter == null )
        return false;
    return mBluetoothAdapter.isEnabled();
}
```

---

#### 4.3.2 enableBluetooth

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	56

This function is used to enable the Bluetooth when Bluetooth is disabled.

---

**public void enableBluetooth( Activity host )**

---

**Parameters**

host

**Return Values**

None

**Remarks**

None

**Example Code**

**JAVA**

**R900Manager.java**

```
private BluetoothAdapter mBluetoothAdapter;
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

public void enableBluetooth( Activity host )
{
    if( mBluetoothAdapter == null || mBluetoothAdapter.isEnabled() == false )
    {
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        host.startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
    }
}
```

---

**4.3.3 queryPairedDevices**

This function is to verify connection with a Bluetooth device.

---

**public Set<BluetoothDevice> queryPairedDevices()**

---


**Parameters**

None

**Return Values**

<p><b>CONFIDENTIAL</b> Document D.O.Tel. All rights of reproduction and disclosure reserved.</p>
--



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	57

None

#### Remarks

None

#### Example Code

##### JAVA

##### R900Manager.java

```
private BluetoothAdapter mBluetoothAdapter;
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

```
public Set<BluetoothDevice> queryPairedDevices()
{
    if( mBluetoothAdapter != null )
        return mBluetoothAdapter.getBondedDevices();
    return null;
}
```

#### 4.3.4 startDiscovery

This function is to start inquiry.

```
public void startDiscovery()
```

#### Parameters

None

#### Return Values

None


#### Remarks

None

#### Example Code

##### JAVA

##### R900Manager.java

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	58

```

public void startDiscovery()
{
    stopDiscovery();
    mBluetoothAdapter.startDiscovery();
}

```

#### 4.3.5 stopDiscovery

This function is to stop inquiry.

```

public void stopDiscovery()

```

##### Parameters

None

##### Return Values

None

##### Remarks

None

##### Example Code

**JAVA**

**R900Manager.java**

```

private BluetoothAdapter mBluetoothAdapter;
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

public void stopDiscovery()
{
    if( mBluetoothAdapter != null && mBluetoothAdapter.isDiscovering() )
        mBluetoothAdapter.cancelDiscovery();
}

```

#### 4.3.6 getBluetoothDevice

This function is to get the information of Bluetooth device.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	59

---

**public BluetoothDevice getBluetoothDevice( String address )**

---

#### Parameters

address

Indicate a MAC address from Bluetooth device.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### R900Manager.java

```
private BluetoothAdapter mBluetoothAdapter;
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

public BluetoothDevice getBluetoothDevice( String address )
{
    if( mBluetoothAdapter != null )
        return mBluetoothAdapter.getRemoteDevice(address);
    return null;
}
```

---

#### 4.3.7 sendData

This function is to send a data to a Bluetooth device.

---

**public void sendData( byte[] bytes )**

---


#### Parameters

bytes

Indicate a data to send.

#### Return Values

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	60

None

#### Remarks

None

#### Example Code

##### JAVA

##### R900Manager.java

```
private BluetoothSocket mBluetoothSocket;
private ConnectedThread mConnectedThread;
mConnectedThread = new ConnectedThread(mBluetoothSocket);

public void sendData( byte[] bytes )
{
    if( mConnectedThread != null )
    {
        Log.d(BluetoothActivity.TAG, "Send : " + new String(bytes));
        mConnectedThread.write(bytes);
    }
    else
    {
        if( mBtEventListener != null )
            mBtEventListener.onBtDataTransException(mBluetoothDevice,
                "[Bluetooth Socket] Write Fail : Bluetooth Thread is not running.");
    }
}
```

#### 4.3.8 isTryingConnect

This function is to check whether trying to connect.

```
public boolean isTryingConnect()
```

#### Parameters

None

#### Return Values

None

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	61

#### Remarks

None

#### Example Code

##### JAVA

##### R900Manager.java

```
public boolean isTryingConnect()
{
    return mConnectThread != null;
}
```

---

#### 4.4 R900Protocol.java.

This code is to control the DOTR-900's protocol.

##### 4.4.1 makeProtocol

This function is to make communication with the DOTR-900's protocol.

---

```
public static final byte[] makeProtocol( String cmd, int[] param )
```

---

#### Parameters

cmd

Indicate a command.

param

Indicate parameters corresponding to a command.

#### Return Values

None

#### Remarks

None

#### Example Code

##### JAVA

##### R900Protocol.java

```
public static final byte[] makeProtocol( String cmd, int[] param )
```

---

**CONFIDENTIAL** Document D.O.Tel. All rights of reproduction and disclosure reserved.

---

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	62

```

{
  StringBuilder protocol = new StringBuilder();
  protocol.append(cmd);

  if( param != null && param.length > 0 )
  {
    for( int i = 0; i < param.length; ++i )
    {
      protocol.append(',');
      if( param[ i ] != SKIP_PARAM )
        protocol.append(param[ i ]);
    }
  }
  return string2bytes( protocol.toString() );
}

```

#### 4.4.2 makeProtocol

This function is to make communication with the DOTR-900's protocol.

```
public static final byte[] makeProtocol( String cmd )
```

#### Parameters

cmd

Indicate a command.

#### Return Values

None

#### Remarks

None

#### Example Code


##### JAVA

##### R900Protocol.java

```

public static final byte[] makeProtocol( String cmd )
{
  StringBuilder protocol = new StringBuilder();
  protocol.append(cmd);
  return string2bytes( protocol.toString() );
}

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	63

#### 4.4.3 makeProtocol

This function is to make communication with DOTR-900's protocol.

---

**public static final byte[] makeProtocol( String cmd, String[] options )**

---

##### Parameters

cmd

Indicate a command.

options

Indicate options corresponding to a command.

##### Return Values

None

##### Remarks

None

##### Example Code

###### JAVA

###### R900Protocol.java

```
public static final byte[] makeProtocol( String cmd, String[] options )
{
    StringBuilder protocol = new StringBuilder();
    protocol.append(cmd);

    if( options != null && options.length > 0)
    {
        for( int i = 0; i < options.length; ++i )
        {
            protocol.append( "," );
            if( options[ i ] != null )
                protocol.append( options[ i ] );
        }
    }
    return string2bytes( protocol.toString() );
}
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	64

#### 4.4.4 makeProtocol

This function is to make communication with DOTR-900's protocol.

---

**public static final byte[] makeProtocol( String cmd, int[] param, String[] options, int[] param2 )**

---

##### Parameters

cmd

Indicate a command.

param

Indicate parameters corresponding to a command.

options

Indicate options corresponding to a command.

param2

Indicate second parameters corresponding to a command.

##### Return Values

None

##### Remarks

None

##### JAVA

##### R900Protocol.java

```


public static final byte[] makeProtocol( String cmd, int[] param, String[] options, int[] param2 )
{
    StringBuilder protocol = new StringBuilder();
    protocol.append(cmd);

    if( param != null && param.length > 0 )
    {
        for( int i = 0; i < param.length; ++i )
        {
            protocol.append(',');
            if( param[ i ] != SKIP_PARAM )
                protocol.append(param[ i ]);
        }
    }

    if( options != null )
    {
        for( int i = 0; i < options.length; ++i )

```



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	65

```

    {
        protocol.append( "," );
        if( options[ i ] != null )
            protocol.append( options[ i ] );
    }
}

if( param2 != null && param2.length > 0 )
{
    for( int i = 0; i < param2.length; ++i )
    {
        protocol.append(',');

        if( param2[ i ] != SKIP_PARAM )
            protocol.append(param2[ i ]);
    }
}

return string2bytes( protocol.toString() );
}

```

#### 4.4.6 makeProtocol

This function is to make communication with DOTR-900's protocol.

```
public static final byte[] makeProtocol( String cmd, int[] param, String option, int[] param2 )
```

##### Parameters

cmd

Indicate a command.

param

Indicate parameters corresponding to a command.

option

Indicate a option corresponding to a command.

param2

Indicate second parameters corresponding to a command.

##### Return Values

None

##### Remarks

None

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	66

## JAVA

### R900Protocol.java

```

public static final byte[] makeProtocol( String cmd, int[] param, String option, int[] param2 )
{
    StringBuilder protocol = new StringBuilder();
    protocol.append(cmd);

    if( param != null && param.length > 0 )
    {
        for( int i = 0; i < param.length; ++i )
        {
            protocol.append(',');
            if( param[ i ] != SKIP_PARAM )
                protocol.append(param[ i ]);
        }
    }

    protocol.append( "," );
    if( option != null )
        protocol.append( option );

    if( param2 != null && param2.length > 0 )
    {
        for( int i = 0; i < param2.length; ++i )
        {
            protocol.append(',');
            if( param2[ i ] != SKIP_PARAM )
                protocol.append(param2[ i ]);
        }
    }

    return string2bytes( protocol.toString() );
}

```

#### 4.4.7 makeProtocol

This function is to make communication with DOTR-900's protocol.

```

public static final byte[] makeProtocol( String cmd, String option, int[] param2 )

```

#### Parameters

cmd

Indicate a command.

options

Indicate options corresponding to a command.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	67

param2

Indicate parameters corresponding to a command.

#### Return Values

None

#### Remarks

None

#### JAVA

##### R900Protocol.java

```

public static final byte[] makeProtocol( String cmd, String option, int[] param2 )
{
    StringBuilder protocol = new StringBuilder();
    protocol.append(cmd);

    protocol.append( "," );
    if( option != null )
        protocol.append( option );

    if( param2 != null && param2.length > 0 )
    {
        for( int i = 0; i < param2.length; ++i )
        {
            protocol.append(',');

            if( param2[ i ] != SKIP_PARAM )
                protocol.append(param2[ i ]);
        }
    }
    return string2bytes( protocol.toString() );
}

```

#### 4.4.8 string2bytes

This function is to convert a data from string type to byte type.

```

public static final byte[] string2bytes( String str )

```

#### Parameters

str

Indicate a string data to convert by byte.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	68

## Return Values

None

## Remarks

None

## JAVA

### R900Protocol.java

```
public static final byte[] string2bytes( String str )
{
    char[] charProtocol = str.toCharArray();
    byte[] byteProtocol = new byte[charProtocol.length + getTypeSize()];
    int index = 0;
    for( int i = 0; i < charProtocol.length; ++i, ++index )
        byteProtocol[ index ] = (byte) ( charProtocol[ i ] & 0xff );

    // ---
    for( int i = 0; i < getTypeSize(); ++i, ++index )
        byteProtocol[ index ] = getType()[ i ];

    return byteProtocol;
}
```

## 4.5 R900RecvPacketParser.java

This code is to parse a data packet.

### 4.5.1 pushPacket

This function is to send a data packet.

```
synchronized public void pushPacket( byte[] buffer, int len )
```


## Parameters

buffer

Indicate a data to send.

len

Indicate length of a data.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	69

#### Return Values

None

#### Remarks

None

#### JAVA

##### R900RecvPacketParser.java

```
synchronized public void pushPacket( byte[] buffer, int len )
{
    if( mCharBuffSize < len )
    {
        mCharBuffSize = ( len << 1 );
        mCharBuff = new char[ mCharBuffSize ];
    }
    for( int i = 0; i < len; ++i )
        mCharBuff[ i ] = (char)( buffer[ i ] & 0xff );
    mPacket.append( mCharBuff, 0, len );
}
```

#### 4.5.2 popPacket

This function is to receive a data packet.

```
synchronized public String popPacket( int offset, int len )
```

#### Parameters

offset

Indicate start location for the received data.

len


Indicate length of the received data.

#### Return Values

None

#### Remarks

None

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	70

## JAVA

### R900RecvPacketParser.java

```
synchronized public String popPacket( int offset, int len )
{
    String pop = mPacket.substring(offset, offset + len);
    mPacket.delete(0, offset + len);
    return pop;
}
```

#### 4.5.3 popPacket

This function is to send a data packet to the DOTR-900.

```
synchronized public String popPacket()
```

#### Parameters

None

#### Return Values

None

#### Remarks

None

## JAVA

### R900RecvPacketParser.java

```
synchronized public String popPacket()
{
    final String STR_PACKET = mPacket.toString();
    final String DELIMETER = R900Protocol.getDelimiter();

    int cmdIndex = STR_PACKET.indexOf("$>");
    if( cmdIndex >= 0 )
    {
        if( STR_PACKET.replace("\n", "").indexOf("$>") == 0 )
            return popPacket(cmdIndex, 2);
    }

    StringTokenizer st = new StringTokenizer(STR_PACKET, DELIMETER);
    if( st.hasMoreTokens() )
```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	71

```

{
    final String str = st.nextToken();
    if( str != null && str.length() > 0 )
    {
        if( str.length() + DELIMITER.length() > mPacket.length() )
            return null;
        if( str.length() == 1 || str.length() > 40 )
            Log.d("kueen108", "Somethins is wrong!!!");
        mPacket.delete(0, Math.min(mPacket.length(), str.length() + DELIMITER.length()));
        return str;
    }
}
return null;
}

```

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	72

## 5. How to install a DEMO

Copy “RFIDUsbHost.apk” to an Android device to install.

After copying application, click it.



Figure 1. Copy an application file.

Click a install button.

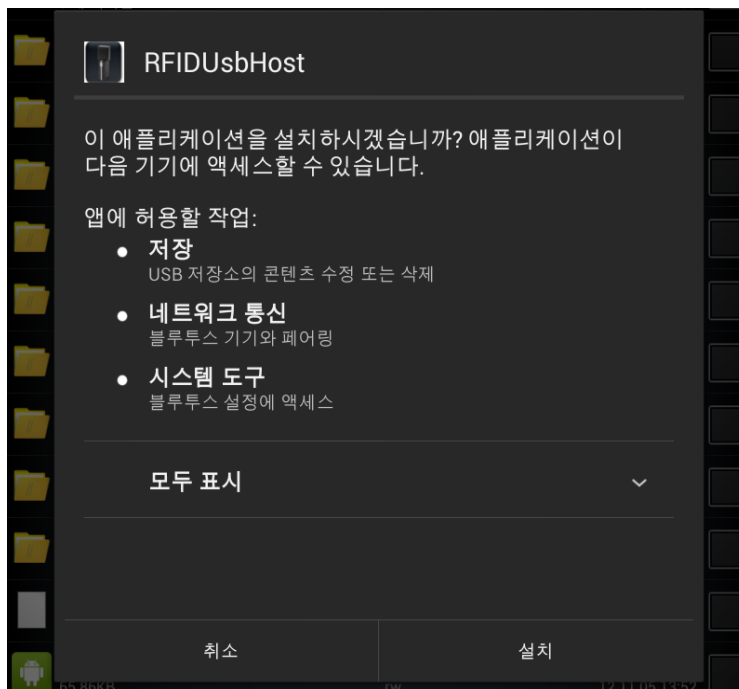


Figure 2. Installation screen for an application.



	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	73

When push a completion button, installation is finished and an application executes when push a open button.

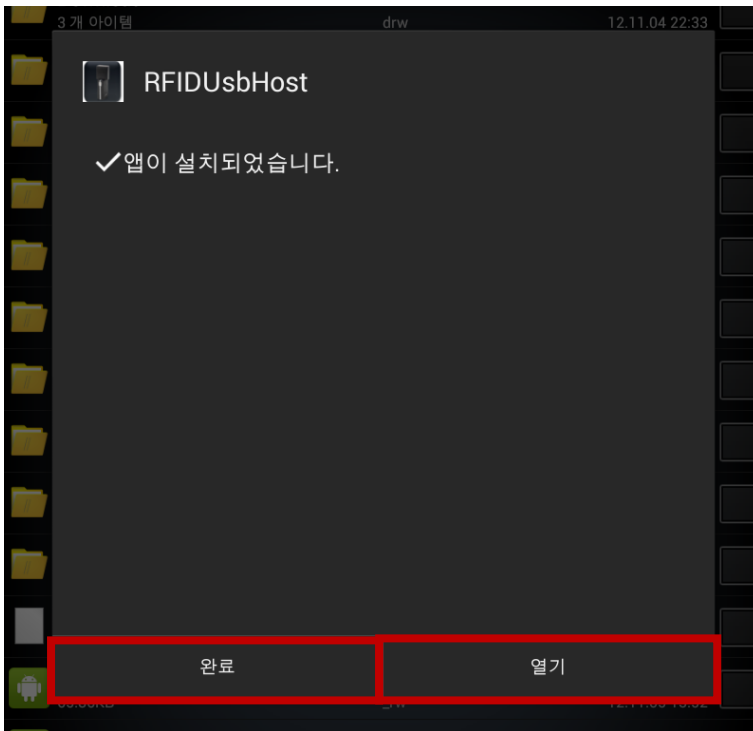


Figure 3. Complete installing “RFIDUsbHost”

When completed installing an application, new icon is created in Apps folder like Figure 4.

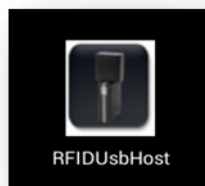



Figure 4. Icon for the RFIDUsbHost application.

## 6. How to use DEMO

This chapter describes main function of the RFIDUsbHost application.

The RFIDUsbHost application supports two modes by USB and Bluetooth.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	74

Initial screen is below when execute the RFIDUsbHost application.



Figure 1. Main screen for the RFIDUsbHost application


When select a Bluetooth button, support connection by Bluetooth.

When select a USB button, support connection by USB.

#### CAUTION :

USB Mode : The DOTR-900 and Android device must be connected by USB cable.

BT Mode : The DoTR-900 and Android device must be released from USB cable.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	75

### USB Mode

In order to use USB mode, an OTG cable must be used.



Figure 2. OTG Cable.

Caution : An Android device have to provide USB Host operation with an OTG cable.

Have to verify that an Android device supports USB Host operation with an OTG cable.

<Table 1> Supporting devices for micro 5pin OTG

Devices	
Samsung	Galaxy S2(SHW-M250S/M250K/M250L), Galaxy S2 LTE(SHW-E110S)
	Galaxy S2 HD LTE(SHV-E120S/E120L), Galaxy Note(SHV-E160L/E160K/E160S)
	Galaxy S3(SHV-M440S), Galaxy S3 LTE(SHV-E210S/E210K/E210L)
	Galaxy Nexus(SHW-M420S/M420K)
Nokia	N810/N8
Motorola	Xoom
Sony	Xperia Arc(only for Global 283 ROM)
Toshiba	Archos G9/TG01
Google	Nexus 7/10

Click a USB button in main screen for a RFIDUsbHost application.

	<h1>DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	76



Figure 3. Link Screen (USB Mode)

An Android device tries to connect to a DOTR-900 by USB when push a “Connect” button.

When push a “Disconnect” button, an Android device releases current USB connection and then goes back to main screen of a RFIDUsbHost application.

If connection completed properly, “Connect OK” pop-up appears.

When connection completed perfectly between a DOTR-900 and an Android device, can use desired operation through “Inventory” or “Access(Read/Write/Lock/Kill)” and “Config” menu.

	<h1 style="text-align: center;">DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	77

## BT Mode

If want to use BT mode, have to release OTG cable between a DOTR-900 and an Android device.

Click a “Bluetooth” button on main screen of a RFIDUsbHost application.

If Bluetooth is not available, pop-up appears about approval for Bluetooth like Figure 4.

If meet this situation, click a “Yes” button for using Bluetooth.

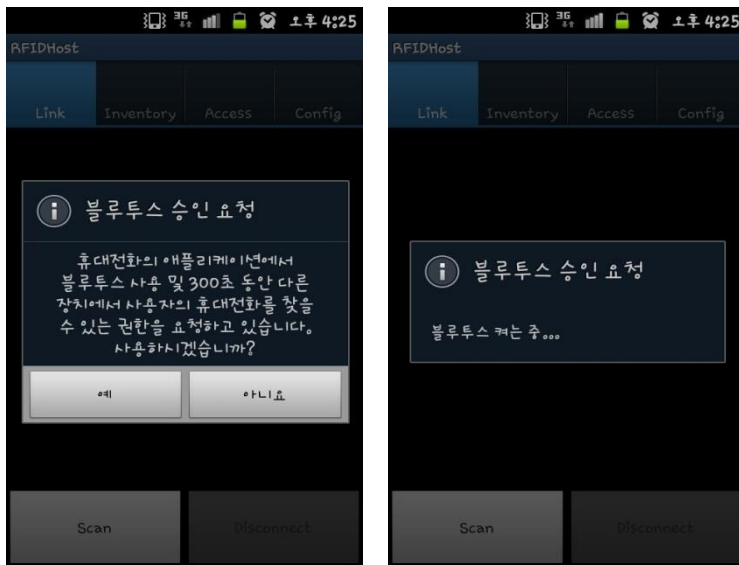


Figure 4. Screen about approval for Bluetooth

After finishing approval for Bluetooth, scan for Bluetooth devices like Figure 5.

If have a previous connection device, a device connects to a reader automatically.

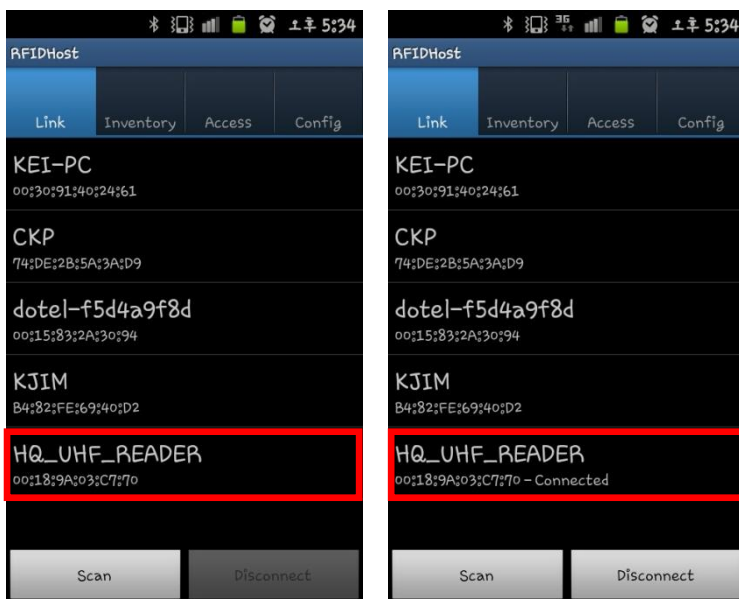


Figure 5. Screen for connection of Bluetooth.



	<h1>DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	79

If click a TAG List Window, menu for list appears like Figure 7.

“Mask” is for setting a mask, “Clear” is for clearing list and “Cancel” is for returning to screen of inventory.

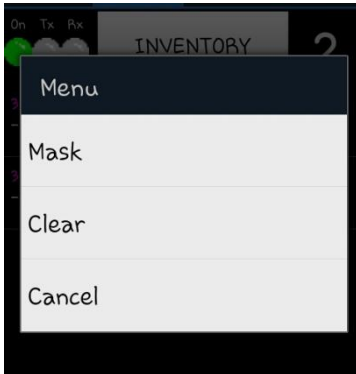


Figure 7. Menu for list

### Screen for Access

This part describes about reading operation in Access.

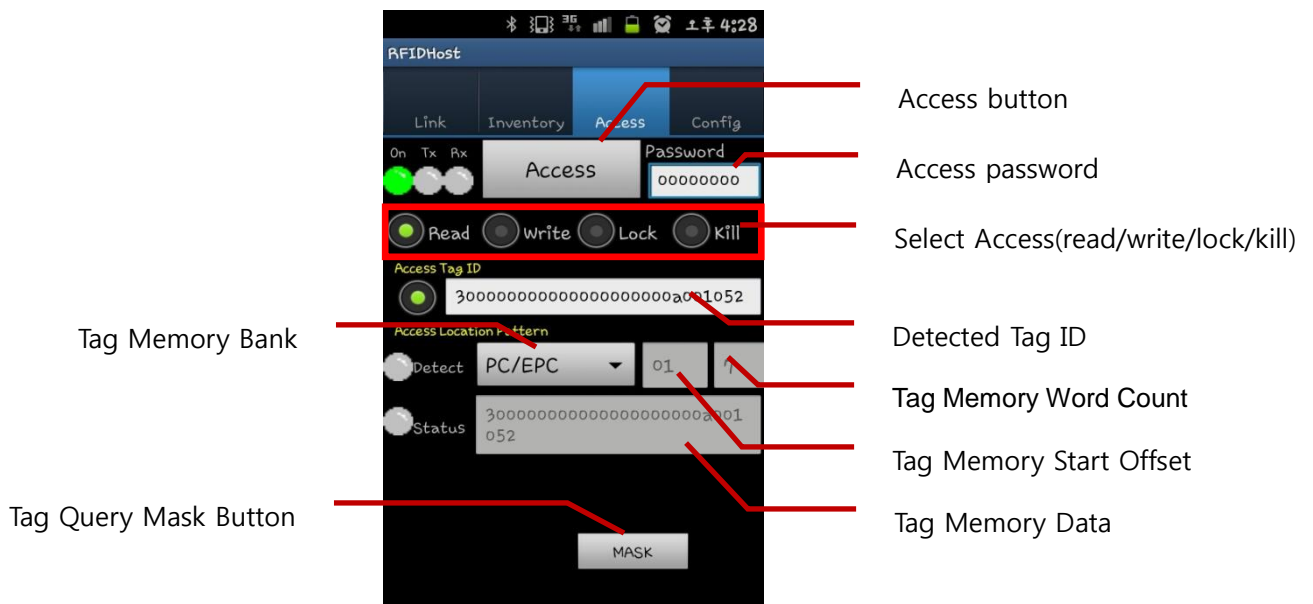


Figure 8. Reading operation in Access

When push a “Access” button, access reading operation begins.


When access reading operation begins, a “Access” button changes to a “STOP” button.

Read a data from memory after detecting TAG ID in access reading operation.

Select a single TAG in screen of CONFIG before executing operation.





	<h1>DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	81

Open state : Stand by mode for Access with inventoried TAG which doesn't have a password of "00000000"

Secured state : Switched status with Access command from Open state.

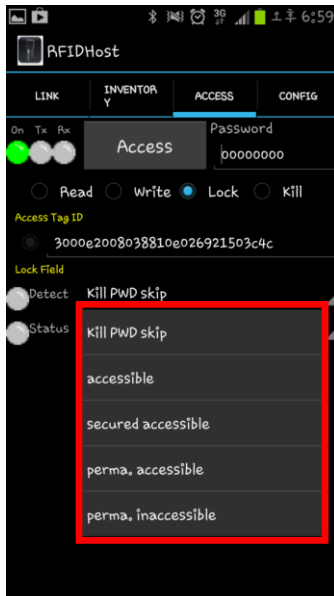


Figure 10. Kill PWD Lock(Access)

"accessible" is to select "kill PWD" to read or change from "Open State" or "Secured State".

"secured accessible" is to select "kill PWD" to read or change from "Secured State".

"perma, accessible" is to select "kill PWD" to read or change from "Open State" or "Secured State".

**Can't change any more after selecting.**

"perma, inaccessible" is that can't select "kill PWD" to read or change from any state.

**Can't change any more after selecting.**

	<h1 style="text-align: center;">DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	82

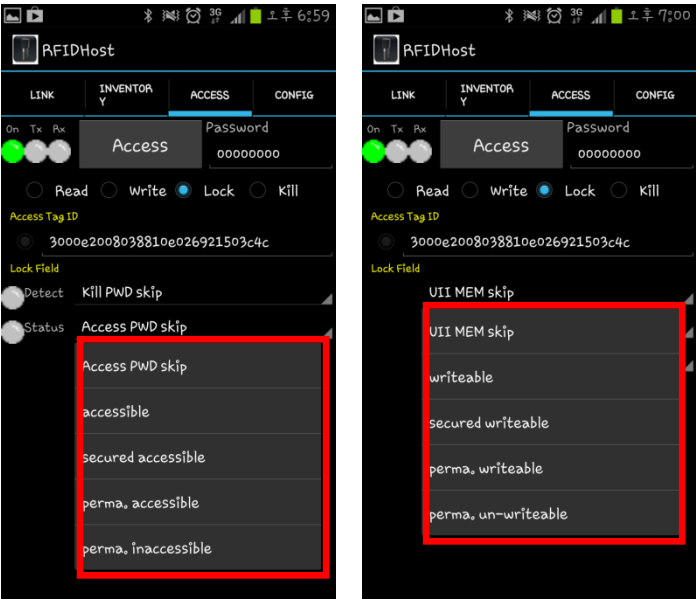


Figure 11. Access PWD and UII MEM Lock(Access)

<Access PWD MEM LOCK>

- “accessible” is to select “lock PWD” to read or change from “Open State” or “Secured State”.
- “secured accessible” is to select “lock PWD” to change from “Secured State”.
- “perma, accessible” is to select “lock PWD” to change from “Open State” or “Secured State”.

**Can’t change any more after selecting.**

- “perma, inaccessible” is that can’t select “lock PWD” to change from any state.

**Can’t change any more after selecting.**


<UII MEM LOCK>

- “writeable” is to select “EPC memory” to change from “Open State” or “Secured State”.
- “secured writeable” is to select “EPC memory” to change from “Secured State”.
- “perma, writeable” is to select “EPC memory” to change from “Open State” or “Secured State”.

**Can’t change any more after selecting.**

- “perma, un-writeable” is that can’t select “EPC memory” to change from “Open State” or “Secured State”.

**Can’t change any more after selecting.**

	<h1 style="text-align: center;">DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	83

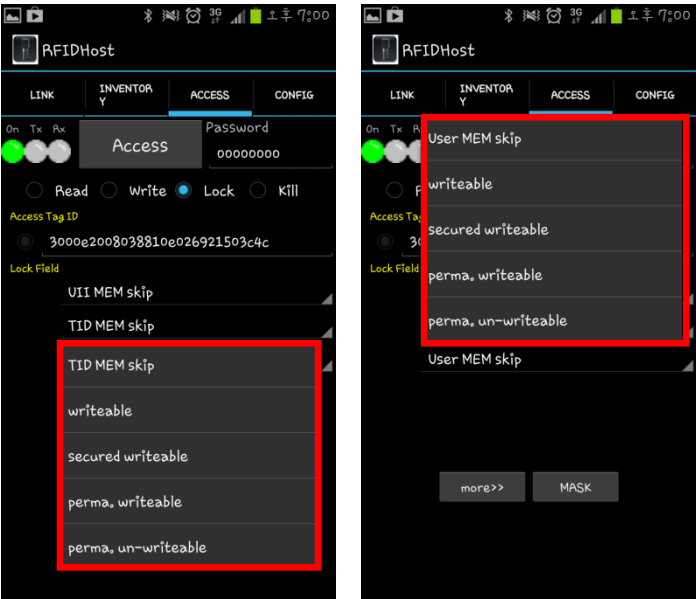


Figure 12. TID MEM and User MEM Lock (Access)

<TID MEM LOCK>

- “writeable” is to select “TID memory” to change from “Open State” or “Secured State”.
- “secured writeable” is to select “TID memory” to change from “Secured State”.
- “perma, writeable” is to select “TID memory” to change from “Open State” or “Secured State”.

**Can’t change any more after selecting.**

- “perma, un-writeable” is that can’t select “TID memory” to change from “Open State” or “Secured State”.

**Can’t change any more after selecting.**

<USER MEM LOCK>

- “writeable” is to select “User memory” to change from “Open State” or “Secured State”.
- “secured writeable” is to select “User memory” to change from “Secured State”.
- “perma, writeable” is to select “User memory” to change from “Open State” or “Secured State”.

**Can’t change any more after selecting.**

- “perma, un-writeable” is that can’t select “User memory” to change from any state.

**Can’t change any more after selecting.**

	<h1>DOTR-900</h1>	Date	2013-08-01
		Rev	1.0
		Page	84

This part describes about killing operation in Access.

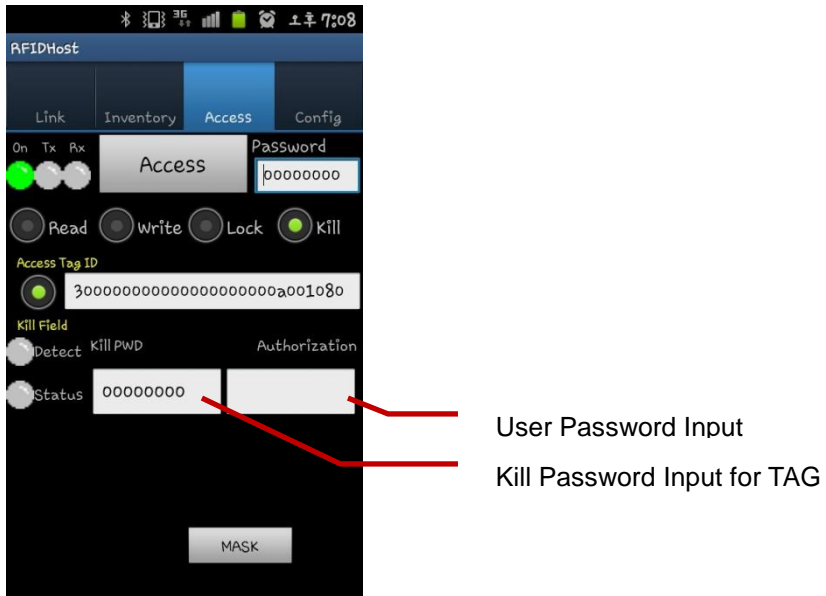


Figure 13. Screen for Kill in Access

Input and execute a Kill Password of TAG. “User Password Input” is to prevent to destroy a TAG by user fault.

In order to destroy a TAG, input “tagkiller” in “User Password Input” and then push a “Access” button

**Need a caution because, the killed TAG can’t be recovered.**

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	85

## Screen for Config

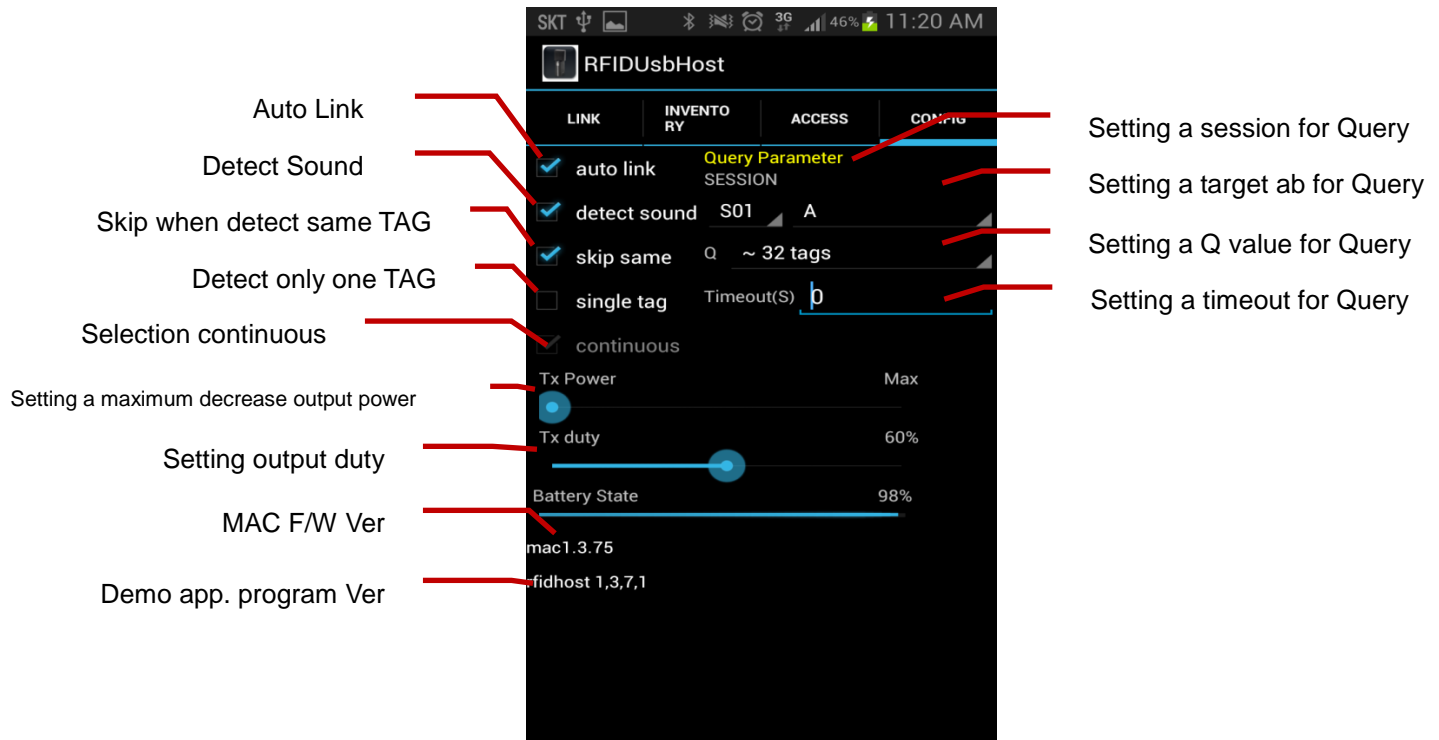


Figure 14. Screen for Config

Set parameters when query TAGs. This parameter value applies to “Inventory” and “Access”.

### **Not support in USB mode.**

A Session can be set to S00, S01, S10 and S11 when query.

A Q value can be set within range, from 1 to 32769 when query.

A target can be set to A, B and AB when query.

If timeout value for Query is 0, there is no timeout. The unit is sec.

A maximum decrease power is 9 dB.


A output duty can be set 20~90% till.

Battery State displays a battery life of DOTR-900.

Select a single tag when running “Access”.

Auto link is to connect automatically when disconnect.

Provide a MAC F/W version.

	<b>DOTR-900</b>	Date	2013-08-01
		Rev	1.0
		Page	86

## Appendix I . How to use Select Mask

Select mask is a command to get response from a TAG which has specific pattern when query a TAG.

Select mask can be set maximum 8 and the set mask applies to “Inventory” or “Access”

ISO 18000-6C/EPC global C1G2 compatibility TAG has 5 Flag in TAG with reference to Query operation.

About 4 sessions, it has the inventoried Flag for each and one selected Flag.

Mask command target code

Code	Target flag
0	Session 0 Inventoried flag
1	Session 1 Inventoried flag
2	Session 2 Inventoried flag
3	Session 3 Inventoried flag
4	Select flag

Mask command action code

Code	Select flag		Inventoried flag	
	Matching tag	Non-matching tag	Matching tag	Non-matching tag
0	SET	RESET	Inventoried -> A	Inventoried -> B
1	SET	unchanged	Inventoried -> A	unchanged
2	unchanged	RESET	unchanged	Inventoried -> B
3	State switched	unchanged	A->B, B->A	unchanged
4	RESET	SET	Inventoried -> B	Inventoried -> A
5	RESET	unchanged	Inventoried -> B	unchanged
6	unchanged	SET	unchanged	Inventoried -> A
7	unchanged	State switched	unchanged	A->B, B->A

Select flag

Desire only one tag type	Action is 0
Exclude only one tag type	Action is 4
Desire several type of tag	Action is 1

※ Because, flag of TAG can maintain its state for a period of time, reset a flag when continuous “Inventory” or “Access”.

※ Set-up mask for Inventory flag must not be used, because it needs special consideration.