



PROFESSIONAL RFID

# MANUAL

API Description

## C9 Black Gun



Version: v1.0

Date: 25.08.2020

## Preface

iDTRONIC GmbH (iDTRONIC) reserves the right to make changes to its products or services or to discontinue any product or service at any time without notice. IDTRONIC provides customer assistance in various technical areas but does not have full access to data concerning the use and applications of customer's products. Therefore, IDTRONIC assumes no liability and is not responsible for customer applications or product or software design or performance relating to systems or applications incorporating IDTRONIC products. In addition, IDTRONIC assumes no liability and is not responsible for infringement of patents and/or any other intellectual or industrial property rights of third parties, which may result from assistance provided by IDTRONIC. IDTRONIC products are not designed, intended, authorized, or warranted to be suitable for life support applications or any other life critical applications that could involve potential risk of death, personal injury or severe property or environmental damage. With the edition of this document, all previous editions become void. Indications made in this manual may be changed without previous notice. Composition of the information in this manual has been done to the best of our knowledge. IDTRONIC does not guarantee the correctness and completeness of the details given in this manual and may not be held liable for damages ensuing from incorrect or incomplete information. Since, despite all our efforts, errors may not be completely avoided, we are always grateful for your useful tips. The installation instructions given in this manual are based on advantageous boundary conditions. IDTRONIC does not give any guarantee promise for perfect function in cross environments. The companies or products mentioned in this document might be brands or brand names of the different suppliers or their subsidiaries in any country. This document may be downloaded onto a computer, stored and duplicated as necessary to support the use of the related IDTRONIC products. Any other type of duplication, circulation, or storage on data carriers in any manner not authorized by IDTRONIC represents a violation of the applicable copyright laws and shall be prosecuted.

### **Safety Instructions / Warning - Read before start-up!**

- The device may only be used for the intended purpose designed by the manufacturer. The operation manual should always be conveniently kept available for each user.
- Unauthorized changes and the use of spare parts and additional devices that have not been sold or recommended by the manufacturer may cause fire, electric shocks or injuries. Such unauthorized measures shall exclude any liability by the manufacturer.
- The liability-prescriptions of the manufacturer in the issue valid at the time of purchase are valid for the device. The manufacturer shall not be held legally responsible for inaccuracies, errors, or omissions in the manual or automatically set parameters for a device or for an incorrect application of a device.
- Repairs may be executed by the manufacturer only.
- Only qualified personnel should carry out installation, operation, and maintenance procedures.
- Use of the device and its installation must be in accordance with national legal requirements and local electrical codes.
- When working on devices the valid safety regulations must be observed.

# CONTENT

<b>Preface .....</b>	<b>2</b>
<b>1 1D/2D Code Reader.....</b>	<b>6</b>
1.1 SetCodedFormat().....	6
1.2 Open().....	6
1.3 Scan() .....	6
1.4 Close().....	6
<b>2 HF-ISO14443A .....</b>	<b>7</b>
2.1 SearchCard14443A().....	7
2.2 Auth14443A ().....	7
2.3 Read14443A () .....	7
2.4 Write14443A () .....	7
<b>3 HF-ISO15693 .....</b>	<b>8</b>
3.1 FindCard15693() .....	8
3.2 GetInformation15693().....	8
3.3 ReadSingleBlock15693 ().....	8
3.4 WriteSingleBlock ().....	8
<b>4 UHF-ISO1800-6C (Impinj R2000) .....</b>	<b>9</b>
4.1 getInstance() .....	9
4.2 getHardware() .....	9
4.3 asyncStartReading() .....	9
4.4 asyncStopReading() .....	9
4.5 setInventoryFilter().....	10
4.6 setCancleInventoryFilter() .....	10
4.7 tagInventoryRealTime() .....	10
4.8 stopTagInventory() .....	11
4.9 tagInventoryByTimer().....	11
4.10 getTagData() .....	11
4.11 getTagDataByFilter .....	12
4.12 writeTagData() .....	12

4.13 writeTagDataByFilter()	13
4.14 writeTagEPC()	13
4.15 writeTagEPCByFilter()	14
4.16 lockTag()	14
4.17 lockTagByFilter()	14
4.18 killTag()	15
4.19 killTagByFilter()	15
4.20 setRegion()	16
4.21 getRegion()	16
4.22 getFrequencyPoints()	16
4.23 setFrequencyPoints()	16
4.24 setPower()	17
4.25 getPower()	17
4.26 setFastMode()	17
4.27 getTemperature()	17
4.28 close()	18
<b>5.0 UHF-ISO1800-6C (ThingMagic M6E Micro)</b>	<b>18</b>
5.1 Power supply	18
5.2 Hardware Module	19
5.3 Connect the hardware	19
5.4 Create read plan	19
5.5 Set read power	20
5.6 Set write power	20
5.7 Set BLF	20
5.8 Set TARI	20
5.9 Set Tagencoding	21
5.10 Set Session	21
5.11 Set Target	21
5.12 Start reading	21
5.13 Get data	22
5.14 Stop reading	22
5.15 Set write data and block	22

5.15	Set the tag to be written .....	23
5.16	Write data.....	23
5.17	Set data read block and length.....	23
5.18	Single read .....	24
<b>6</b>	<b>Fingerprint .....</b>	<b>24</b>
6.1	Open().....	24
6.2	DetectFinger() .....	24
6.3	GetFingerprintImage() .....	24
6.4	GetChara() .....	25
6.5	RegisterFingerprint ().....	25
6.6	SearchFingerprint() .....	25
6.7	MatchFingerprint() .....	25
6.8	GetFreeAddress() .....	26
6.9	EmptyAllChara() .....	26
6.10	EmptyChara().....	26
<b>7</b>	<b>LF125KManager.....</b>	<b>26</b>
7.1	Open().....	26
7.2	Start() .....	26
7.3	Pause() .....	27
7.4	Close().....	27
<b>8</b>	<b>Appendix.....</b>	<b>27</b>
8.1	Key Codes.....	27
8.2	SerialPort .....	27

# 1 1D/2D Code Reader

## 1.1 SetCodedFormat()

Function	void SetCodedFormat(String format)
Description	Set encoding format
Parameter	Format
Return	void

*Note: Format of "UTF-8" by default.*

## 1.2 Open()

Function	boolean Open(Handler handler)
Description	Load the 1D/2D Scanner and Start Receiving Thread
Parameter	Handler
Return	True: success False:failure

*Note: Int the Handler: msg.what == Barcode1DManager.Barcode1D  
String data = msg.getData().getString("data");*

## 1.3 Scan()

Function	void Scan()
Description	Start the 1D/2D Scanner to Scan Barcode
Parameter	void
Return	void

## 1.4 Close()

Function	boolean Close()
Description	Unload the 1D/2D Scanner and Interrupt Receiving Thread
Parameter	void
Return	True:success False:failure

*Note: Close the hardware module when the process is finished.*

## 2 HF-ISO14443A

### 2.1 SearchCard14443A()

Function	byte[] SearchCard14443A(Error error)
Description	Used to search iso14443A card
Parameter	Error
Return	Byte[]: ISO14443A card uid Null: no card

*Note: HfError error = new HfError(); The following error description is the same.*

### 2.2 Auth14443A ()

Function	int auth14443A(int authType, byte[] access, byte[] uid, int sector, Error error)
Description	auth iso14443A card
Parameter	authType , password , uid , sector number , error
Return	0: auth success Other: error code

### 2.3 Read14443A ()

Function	byte[] read14443A(int block, Error error)
Description	Read ISO14443A card data
Parameter	Block number, error
Return	Byte[]: block data Null: read failure

*Note: Block number ranges from block 0 to block 63;*

### 2.4 Write14443A ()

Function	int write14443A(byte[] writeData, int block, Error error)
Description	Write iso14443A card data
Parameter	Byte[] data, block number, error
Return	0: write success Other: write failure, error code

*Note: Length of the data should be 16 bytes.*

## 3 HF-ISO15693

### 3.1 FindCard15693()

Function	List<Iso15693InventoryInfo> findCard15693(Error error)
Description	Used to find ISO15693 card
Parameter	Error
Return	Iso15693InventoryInfo list, Iso15693InventoryInfo contain flag , dsfid and uids

### 3.2 GetInformation15693()

Function	Iso15693CardInformation getInformation15693(byte[] uid, int flag ,Error error)
Description	To get ISO15693 card system information
Parameter	Card uid , flag , Error
Return	Iso15693CardInformation Null: get failure

### 3.3 ReadSingleBlock15693 ()

Function	byte[] readSingleBlock15693(byte[] uid, int flag,int block, Error error)
Description	To read ISO15693 card single block data
Parameter	Card uid , flag , block number , Error
Return	Byte[] block data Null: read failure

### 3.4 WriteSingleBlock ()

Function	int writeSingleBlock(byte[] uid,int flag, int block, byte[] writeData, Error error)
Description	To write ISO15693 card single block data
Parameter	Card uid , flag , block number , data , Error
Return	0: write success Other: write failure



## 4 UHF-ISO1800-6C (Impinj R2000)

### 4.1 getInstance()

Function	UHFRManager getInstance()
Description	Get UHF instance, and open the hardware module
Parameter	Null
Return	UHFRManager instance

### 4.2 getHardware()

Function	String getHardware()
Description	Get hardware version
Parameters	Null
Return	Hardware version string Null failure

### 4.3 asyncStartReading()

Function	READER_ERR asyncStartReading()
Description	Start reading of multiple mode
Parameters	Null
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure

### 4.4 asyncStopReading()

Function	READER_ERR asyncStopReading()
Description	Stop reading of multiple mode, use together with asyncStartReading()
Parameters	Null
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure

#### 4.5 setInventoryFilter()

Function	boolean setInventoryFilter()
Description	Set inventory filter
	byte[] fdata, data to be filtered int fbank, bank of data to be filtered 1 :EPC, 2: TID , 3: USER int fstartaddr, starting address, unit of word boolean matching
Parameters	true: inventory the matched tag; false: inventory the non-matching tag
Return	True for success False for failure

#### 4.6 setCancelInventoryFilter()

Function	boolean setInventoryFilter()
Description	Set inventory filter
	byte[] fdata, data to be filtered int fbank, bank of data to be filtered 1 :EPC, 2: TID , 3: USER int fstartaddr, starting address, unit of word boolean matching
Parameters	true: inventory the matched tag; false: inventory the non-matching tag
Return	True for success False for failure

#### 4.7 tagInventoryRealTime()

Function	public List<TAGINFO> tagInventoryRealTime()
Description	Inventory in real-time and show the tag list, use after asyncStartReading()
Parameters	Null
Return	List<TAGINFO> Tag list Null

#### 4.8 stopTagInventory()

Function	boolean stopTagInventory()
Description	Stop inventory
Parameters	Null
Return	True for success False for failure

#### 4.9 tagInventoryByTimer()

Function	List<TAGINFO> tagInventoryByTimer(short readtime)
Description	Inventory by timer
Parameters	Short readtime, time of single inventory with unit of ms
Return	List<TAGINFO> Tag list Null: reading failure

#### 4.10 getTagData()

Function	READER_ERR getTagData(int mbank, int startaddr, int len, byte[] rdata, byte[] password, short timeout)
Description	Get tag data
Parameters	int mbank, bank to be read, 0:RESERVED, 1:EPC,2:TID, 3:USER int startaddr, starting address of the tag with unit of word int len, length of the tag byte[] rdata, read data, same length as len byte[] password, access password, 4 bytes short timeout, timeout value with unit of ms
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure

## 4.11 getTagDataByFilter

Function	byte[] getTagDataByFilter(int mbank, int startaddr, int len, byte[] password, short timeout, byte[] fdata, int fbank, int fstartaddr, boolean matching)
Description	Read specified tag by filter
Parameters	<p>int mbank, memory bank, 0:RESERVED, 1:EPC, 2:TID, 3:USER</p> <p>int startaddr, starting address with unit of word</p> <p>int len, length of the data to be read</p> <p>byte[] password, access password, 4 bytes</p> <p>short timeout, timeout value with unit of ms</p> <p>byte[] fdata, filtered data</p> <p>int fbank, filtered bank, 1 :EPC, 2: TID ,3: USER</p> <p>int fstartaddr, starting address with unit of word</p> <p>boolean matching or not, true: read the matching tag; false: read the non-matching tag</p>
Return	Byte[] Success Null Failure

## 4.12 writeTagData()

Function	READER_ERR writeTagData(char mbank, int startaddress, byte[] data, int datalen, byte[] accesspasswd, short timeout)
Description	Write data
Parameters	<p>int mbank, memory bank to be written, 0:RESERVED, 1:EPC, 2:TID, 3:USER</p> <p>int startaddress, starting address with unit of word</p> <p>byte[] data, data to be written</p> <p>int datalen, length of the data with unit of word</p> <p>byte[] password, access password, 4 bytes</p> <p>short timeout, timeout value with unit of ms</p>
Return	<p>READER_ERR</p> <p>Success: Reader.READER_ERR.MT_OK_ERR</p> <p>Failure: error information</p>

#### 4.13 writeTagDataByFilter()

Function	READER_ERR writeTagDataByFilter(char mbank, int startaddress, byte[] data, int datalen, byte[] accesspasswd, short timeout, byte[] fdata, int fbank, int fstartaddr, boolean matching)
Description	Write data to the specified tag by filter
Parameters	int mbank, memory bank of data to be written, 0:RESERVED, 1:EPC, 2:TID, 3:USER int startaddress, starting address with unit of word byte[] data, data to be written int datalen, length of the data to be written with unit of word byte[] password, access password, 4 bytes short timeout, timeout value with unit of ms byte[] fdata, filtered data int fbank, filtered bank 1:EPC,2:TID,3:USER int fstartaddr, starting address with unit of word boolean matching or not, true: write data to the matching tag; false: write data to the non-matching tag
Return	READER_ERR Success : Reader.READER_ERR.MT_OK_ERR Failure : error information

#### 4.14 writeTagEPC()

Function	READER_ERR writeTagEPC(byte[] data, byte[] accesspwd, short timeout)
Description	Write EPC
Parameters	byte[] data, EPC data to be written byte[] accesspwd, access password, 4 bytes short timeout, timeout value with unit of ms
Return	READER_ERR Success : Reader.READER_ERR.MT_OK_ERR Failure: error list

#### 4.15 writeTagEPCByFilter()

Function	READER_ERR writeTagEPCByFilter(byte[] data, byte[] accesspwd, short timeout, byte[] fdata, int fbank, int fstartaddr, boolean matching)
Description	Write EPC of specified tag by filter
Parameters	byte[] data, EPC data to be written byte[] accesspwd, access password, 4 bytes short timeout, timeout value with unit of ms byte[] fdata, filtered data int fbank, filtered bank 1 :EPC,2: TID ,3: USER int fstartaddr, starting address with unit of word boolean matching or not, true: write data to the matching tag; false: write data to the non-matching tag
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure: error information

#### 4.16 lockTag()

Function	READER_ERR lockTag(Lock_Obj lockobject, Lock_Type locktype, byte[] accesspasswd, short timeout)
Description	Lock tag
Parameters	Lock_Obj lockobject, lock object including access password, kill password, EPC bank and USER bank, please refer to the demo source codes Lock_Type locktype, lock type, including lock/unlock, permanent lock, please refer to the demo source codes byte[] accesspasswd, access password, 4 bytes short timeout, timeout value with unit of ms
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure: error information

#### 4.17 lockTagByFilter()

Function	READER_ERR lockTagByFilter(Lock_Obj lockobject, Lock_Type locktype, byte[] accesspasswd, short timeout, byte[] fdata, int fbank, int fstartaddr, boolean matching)
Description	Lock tag by filter
Parameters	Lock_Obj lockobject, lock object including access password, kill password, EPC bank and USER bank, please refer to the demo source codes

	Lock_Type locktype, lock type, including lock/unlock, permanent lock, please refer to the demo source codes byte[] accesspasswd, access password, 4 bytes short timeout, timeout value with unit of ms byte[] fdata, filtered data int fbank, filtered bank, 1 :EPC,2: TID ,3: USER int fstartaddr, starting address with unit of word boolean matching or not, true: lock the tag matching the filter; false: lock the tag non-matching the filter READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure: error information
Return	

#### 4.18 killTag()

Function	READER_ERR killTag(byte[] killpasswd, short timeout) Kill the tag, please be aware about the operating since the tag would not work after being killed Tag cannot be killed with password of 0
Description	
Parameters	byte[] killpassword, kill password short timeout, timeout value with unit of ms READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure;
Return	

#### 4.19 killTagByFilter()

Function	READER_ERR killTagByFilter(byte[] killpasswd, short timeout, byte[] fdata, int fbank, int fstartaddr, boolean matching)
Description	Kill the specified tag by filter
Parameters	byte[] killpasswd, kill password short timeout, timeout value with unit of ms byte[] fdata, data to be filtered int fbank, bank of data to be filtered 1 :EPC,2: TID ,3: USER int fstartaddr, starting address with unit of word boolean matching or not; true: kill the tag that matches the filter data, false: kill the tag that does not match the filter READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure:
Return	

## 4.20 setRegion()

Function	READER_ERR setRegion(Region_Conf region)
Description	Set the frequency region
Parameters	Region_Conf region, including CHN, USA, Korea, EU. Failure indicates the frequency is not supported
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure

## 4.21 getRegion()

Function	Region_Conf getRegion()
Description	Get the current frequency region
Parameters	Null
Return	Region_Conf, please refer to the demo source codes

## 4.22 getFrequencyPoints()

Function	int[] getFrequencyPoints()
Description	Get the frequency points
Parameters	Null
Return	Int[] Frequency points with unit of kHz

## 4.23 setFrequencyPoints()

Function	READER_ERR setFrequencyPoints(int[] frequencyPoints)
Description	Set frequency points, frequency points can be obtained by getFrequencyPoints()
Parameters	int[] frequencyPoints with unit of kHz
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR Failure



#### 4.24 setPower()

Function	READER_ERR setPower(int readPower, int writePower)
Description	Set the power value
Parameters	int readPower, range: 5~33 int writePower, range: 5~33
Return	READER_ERR Success: Reader.READER_ERR.MT_OK_ERR

#### 4.25 getPower()

Function	int[] getPower()
Description	Get the power
Parameters	Null
Return	Int[] success, int[0] as reading power, int[1] as writing power Null failure

#### 4.26 setFastMode()

Function	READER_ERR setFastMode()
Description	Set fast/multiple reading mode with the max power
Parameters	Null
Return	List<EPCDataModel>: EPC data list Null

#### 4.27 getTemperature()

Function	int getTemperature()
Description	Get the module chip temperature
Parameters	Null
Return	int >0 as success, <0 as failure

## 4.28 close()

Function	Boolean close()
Description	Close the hardware connection
Parameters	Null
Return	True for success; Fail for failure

**For more details, please refer to the demo source codes.**

## 5.0 UHF-ISO1800-6C (ThingMagic M6E Micro)

### API Jar

aeitagdecoder.jar

d2xx.jar

gson-2.8.5.jar

jmdns-3.4.1.jar

ltkjava-1.0.0.6.jar

uhf\_magic\_1.1.jar

## 5.1 Power supply

Power on:

Function	xt_PowerOn() serialPort.rfid_poweron();	
Description	Power on the UHF module	
Parameter	None	
Return	True or False	

Power off:

Function	public void close()
Description	Power off the UHF module
Parameter	None
Return	Null

## 5.2 Hardware Module

### Create Reader Object

Function	Reader.create(query);
Description	Create reader object
Parameter	Serial port address
Return	Reader object instance

## 5.3 Connect the hardware

Function	reader.connect();
Description	Connect the reader
Parameter	None
Return	Null

## 5.4 Create read plan

Function	SimpleReadPlan plan = new SimpleReadPlan(antennaList, TagProtocol.GEN2, null, null, 1000);
Description	Create read plan by paramSet
Parameter	antennaList --- int[] antennaList = {1}; protocol --- agProtocol.GEN2 filter --- op ---
Return	SimpleReadPlan object

## 5.5 Set read power

Function	<code>reader.paramSet(TMConstants.TMR_PARAM_RADIO_READPOWER, readpower);</code>
Description	Set read power
Parameter	Int
Return	Null

## 5.6 Set write power

Function	<code>reader.paramSet(TMConstants.TMR_PARAM_RADIO_WRITEPOWER, writepower);</code>
Description	Set write power
Parameter	Int
Return	Null

## 5.7 Set BLF

Function	<code>Gen2.LinkFrequency blf = Gen2.LinkFrequency.LINK250KHZ; reader.paramSet(TMConstants.TMR_PARAM_GEN2_BLF, blf);</code>
Description	Set BLF
Parameter	Gen2.LinkFrequency
Return	Null

## 5.8 Set TARI

Function	<code>Gen2.Tari tari = Gen2.Tari.TARI_6_25US; reader.paramSet(TMConstants.TMR_PARAM_GEN2_TARI, tari);</code>
Description	Set TARI
Parameter	Gen2.TagEncoding
Return	Null

## 5.9 Set Tagencoding

Function	<code>Gen2.TagEncoding tagEncoding = Gen2.TagEncoding.FM0; reader.paramSet(TMConstants.TMR_PARAM_GEN2_TAGENCODING, tagEncoding);</code>
Description	Set Tagencoding
Parameter	Gen2.TagEncoding
Return	Null

## 5.10 Set Session

Function	<code>Gen2.Session session = Gen2.Session.S0; reader.paramSet(TMConstants.TMR_PARAM_GEN2_SESSION, session);</code>
Description	Set Session
Parameter	Gen2.Session
Return	Null

## 5.11 Set Target

Function	<code>Gen2.Target target = Gen2.Target.A; reader.paramSet(TMConstants.TMR_PARAM_GEN2_TARGET, target);</code>
Description	Set Target
Parameter	Gen2.Target
Return	Null

## 5.12 Start reading

Function	<code>r.startReading();</code>
Description	Start reading
Parameter	None
Return	Callback Readlistener

### 5.13 Get data

Function	<pre>ReadListener rl = new PrintListener(); r.addReadListener(rl); static class PrintListener implements ReadListener {     public void tagRead(Reader r, TagReadData tr)     {         System.out.println("Background read: " + tr.toString());     } }</pre>
Description	Get the tag data
Parameter	None
Return	Tag data

### 5.14 Stop reading

Function	<pre>r.stopReading();</pre>
Description	Stop reading
Parameter	None
Return	Null

### 5.15 Set write data and block

Function	<pre>Gen2.BlockWrite tagop = new Gen2.BlockWrite(Gen2.Bank.USER, 0, (byte) 2, writeData);</pre>
Description	Set block information of the data to be written
Parameter	Parameter 1: Block Parameter 2: starting address Parameter 3: data length Parameter 4: write data
Return	Gen2.BlockWrite object

### 5.15 Set the tag to be written

Function	Gen2.Select select = new Gen2.Select(false, Gen2.Bank.EPC, 32, 96, tagReads[0].getTag().epcBytes());
Description	Select the tag to be written
Parameter	Parameter 1: false by default Parameter 2: select bank Parameter 3: starting address with unit of bit Parameter 4: data size with unit of bit Parameter 5: data
Return	Gen2.Select object

### 5.16 Write data

Function	r.executeTagOp(tagop, select);
Description	Execute write data
Parameter	Gen2.BlockWrite Gen2.Select
Return	Null

### 5.17 Set data read block and length

Function	Gen2.ReadData readOp = new Gen2.ReadData(Gen2.Bank.USER, 0, (byte) 0x02);
Description	Set data read block and length
Parameter	Parameter 1: set the block Parameter 2: starting address Parameter 3: data length
Return	Gen2.ReadData

## 5.18 Single read

Function	r.executeTagOp(tagop, select);
Description	Read data of specified block
Parameter	Parameter 1: Gen2.ReadData object Parameter 2: Gen2.Select object
Return	Data read

# 6 Fingerprint

## 6.1 Open()

Function	boolean Open()
Description	Open fingerprint device power
Parameter	boolean isRoot, Context context
Return	True: open success False: open failure

*Note: If you have already get root permissions ,isRoot = true, else isRoot = false.*

## 6.2 DetectFinger()

Function	boolean DetectFinger()
Description	Detect whether the finger on the sensor
Parameter	void
Return	True: finger exist False: finger inexistence

## 6.3 GetFingerprintImage()

Function	Bitmap GetFingerprintImage()
Description	Get fingerprint image bitmap
Parameter	void
Return	Bitmap: get fingerprint success Null: get fingerprint image failure

*Note: Place the finger on the sensor properly during the process.*



## 6.4 GetChara()

Function	byte[] GetChara()
Description	Get fingerprint characteristic value
Parameter	void
Return	Byte[]: Characteristic value Null: get fingerprint characteristic value failure

*Note: Place the finger on the sensor properly during the process.*

## 6.5 RegisterFingerprint ()

Function	int RegisterFingerprint(byte[] chara_1,byte[] chara_1_more,int min_score)
Description	Save fingerprint characteristic value in flash
Parameter	characteristic value_1, characteristic value_2, minimum score
Return	Int >=0: address ID of fingerprint characteriistic value in (register success ) Int<0: error code (register failure)

*Note: characteristic value\_1 and characteristic value\_2 must be from the same finger.*

*Min\_score should be greater than 50;*

## 6.6 SearchFingerprint()

Function	int SearchFingerprint(byte[] chara)
Description	Search the fingerprint characteristic value from flash memory
Parameter	Finger print characteristic value
Return	Int >=0: address ID of the fingerprint characteristic value in the flash(search success ) Int<0: error code (search failure)

*Note: Parameters should be obtained dynamically from GetChara()*

## 6.7 MatchFingerprint()

Function	int MatchFingerprint(byte[] chara_1,byte[] chara_2)
Description	Compare fingerprint characteristic value_1 and fingerprint characteristic value_2 to judge whether are they the same fingerprint ?
Parameter	characteristic value 1, characteristic value 2
Return	Match score

*Note: Score value larger than 50 indicates the same finger*

## 6.8 GetFreeAddress()

Function	int GetFreeAddress()
Description	Get free address
Parameter	void
Return	Free address ID

## 6.9 EmptyAllChara()

Function	boolean EmptyAllChara()
Description	Delete all fingerprint characteristic values than have registered in the flash
Parameter	void
Return	True : delete success False: delete failure

## 6.10 EmptyChara()

Function	boolean EmptyChara(int addressID)
Description	Delete the fingerprint characteristic value by address than has registered in the flash
Parameter	Address
Return	True : delete success False: delete failure

# 7 LF125KManager

## 7.1 Open()

Function	boolean Open(Handler mHandler)
Description	Open device
Parameter	Handler
Return	void

*Note: Int the Handler: msg.what == LfHdxManager.LF) {  
Bundle bundle = msg.getData();  
String data = bundle.getString("data");*

## 7.2 Start()

Function	void start()
Description	Start read card or continue read card after pause
Parameter	void
Return	void

### 7.3 Pause()

Function	void Pause()
Description	Pause read card after start
Parameter	void
Return	void

### 7.4 Close()

Function	boolean Close()
Description	Close device
Parameter	void
Return	void

## 8 Appendix

### 8.1 Key Codes

1. F1: 131
2. F2: 132
3. Left button: 133
4. Right button: 135

### 8.2 SerialPort

#### Open Serial Port

Function	SerialPort(int port, int baudrate, int flags)
Description	Open serial port
Parameter	Port number, Port baud rate Flags =0
Return	Serial port class instance

*Note: Send and receive the data after opening serial port.*

#### Close Serial Port

Function	close(int port)
Description	Close serial port
Parameter	Port number
Return	void

*Note: Power management of serial port is packaged in the SerialPort.class, please reference to SerialPortActivity.class for more details.*