Advanced Card Systems Ltd.
Card & Reader Technologies

AMR220-C1

ACS Secure Bluetooth® mPOS Reader

Reference Manual V1.04

info@acs.com.hk
www.acs.com.hk

Revision History

Release Date	Revision Description	Version Number
2018-02-08	• Initial Release	1.00
2018-03-02	• Updated Section 2.00: Features • Updated Section 3.1.2 photos • Added Section 3.1.2.3: Device Reset Pinhole	1.01
2018-07-19	• Updated Section 3.1.1.2 Battery Life	1.02
2018-09-04	• Updated Section 2.0 Features	1.03
2019-06-17	• Updated Reader Marketing Name • Updated Section 5.3.1: Get Firmware Version • Updated Section 5.3.3: Antenna Field Control • Updated Section 5.3.4: Automatic PICC Polling • Updated Section 5.3.5: PICC Operating Parameter • Updated Section 5.3.7: LED Control	1.04

Table of Contents

List of Figures

List of Tables

1.0. Introduction

The AMR220-C1 ACS Secure Bluetooth® mPOS Reader communicates with smart devices via Bluetooth® technology. With its compliance to ISO 7816 and ISO 14443, it supports both contact and contactless smart cards. Moreover, it further extends card support and strengthens the mobile reader product line's salability in the payment industry, with the additional compliance to EMV® Levels 1 & 2, Mastercard® Contactless (formerly MasterCard PayPass), and Visa® Contactless.

Target customers include micro merchants, mobile merchants (example: catering, food trucks, express delivery companies), and retail merchants.

1.1. Symbols and Abbreviations

Abbreviation	Description
AC	Application Cryptogram
AID	Application Identifier
AIP	Application Interchange Profile
AOSA	Available Offline Spending Amount
APDU	Application Protocol Data Unit
ATC	Application Transaction Counter
BT	Bluetooth
BLE	Bluetooth Low Energy
CA	Certification Authority
CED	Customer Exclusive Data
CID	Cryptogram Information Data
CVM	Cardholder Verification Method
CVR	Card Verification Results
DD	Discretionary Data
DF	Dedicated File
FFI	Form Factor Indicator
FW	Firmware
IAD	Issuer Application Data
IFD	Interface Device
JCB	Japan Credit Bureau
PAN	Primary Account Number

Abbreviation	Description
PBOC	People's Bank of China specifications
PCD	Proximity Coupling Device
PIN	Personal Identification Number
POS	Point of Sale
PSN	Application PAN Sequence Number
RID	Registered Application Provider Identifier
QPBOC	Quick PBOC (The Chinese counterpart of contact-less EMV)
TAC	Terminal Action Code
TTQ	Terminal Transaction Qualifiers
TVR	Terminal Verification Results

Table 1: Symbols and Abbreviations

2.0. Features

- USB Full Speed Interface
- Bluetooth® Interface
- Plug and Play – CCID support brings utmost mobility
- Smart Card Reader:
  - Contactless Interface:
    - Read/Write speed of up to 848 Kbps
    - Built-in antenna for contactless tag access, with reading distance of up to 50 mm (depending on tag type)
    - Supports ISO 14443 Part 4 Type A and B cards, MIFARE®, FeliCa, and all 4 types of NFC (ISO/IEC 18092) tags
    - Supports Mastercard® Contactless and Visa payWave® compliant cards
    - Built-in anti-collision feature (only one tag is accessed at any time)
    - NFC Mode Supported:
      - Card reader/writer mode
  - Contact Interface:
    - Read/Write speed of up to 600 Kbps
    - Supports ISO 7816 Class A, B, and C (5 V, 3 V, 1.8 V) full-sized cards
    - Supports microprocessor cards with T=0 or T=1 protocol
    - Supports PPS (Protocol and Parameters Selection)
    - Features Short Circuit Protection
- Application Programming Interface:
  - Supports PC/SC
  - Supports CT-API (through wrapper on top of PC/SC)
- Built-in Peripherals:
  - LEDs:
    - Four User-controllable single-color LED (Green)
    - One Charging Status LED (Red)
    - One Bluetooth Status LED (Blue)
  - Buttons:
    - Power Switch
    - Bluetooth Switch
  - User-controllable speaker (audio tone indication)
- Supports several cryptographic algorithms (Upon Request) such as AES, DES, and 3DES
- USB Firmware Upgradeability[1]
- Supports Android™ 4.4 and later[2]
- Supports iOS 8.0 and later[3]
- Compliant with the following standards:
  - EN 60950/IEC 60950

[1] *Applicable under PC-linked mode*
[2] *Uses an ACS-defined Android Library*
[3] *Uses an ACS-defined iOS Library*

- ISO 7816
- ISO 14443
- ISO 18092
- EMV® Levels 1 and 2
- Mastercard® Contactless
- Visa payWave®
- Bluetooth®
- PC/SC
- CCID
- CE
- FCC
- RoHS 3
- REACH
- MIC (Japan)
- Microsoft® WHQL

3.0. Architecture

3.1. Hardware Design

The Cortex M3 grade main processor is used for communication with tablets or PCs via BT interface or USB interface. It also controls the peripherals and ICC communication. The NFC chip acts as a transceiver to build an RF channel between a contactless tag and the main processor.

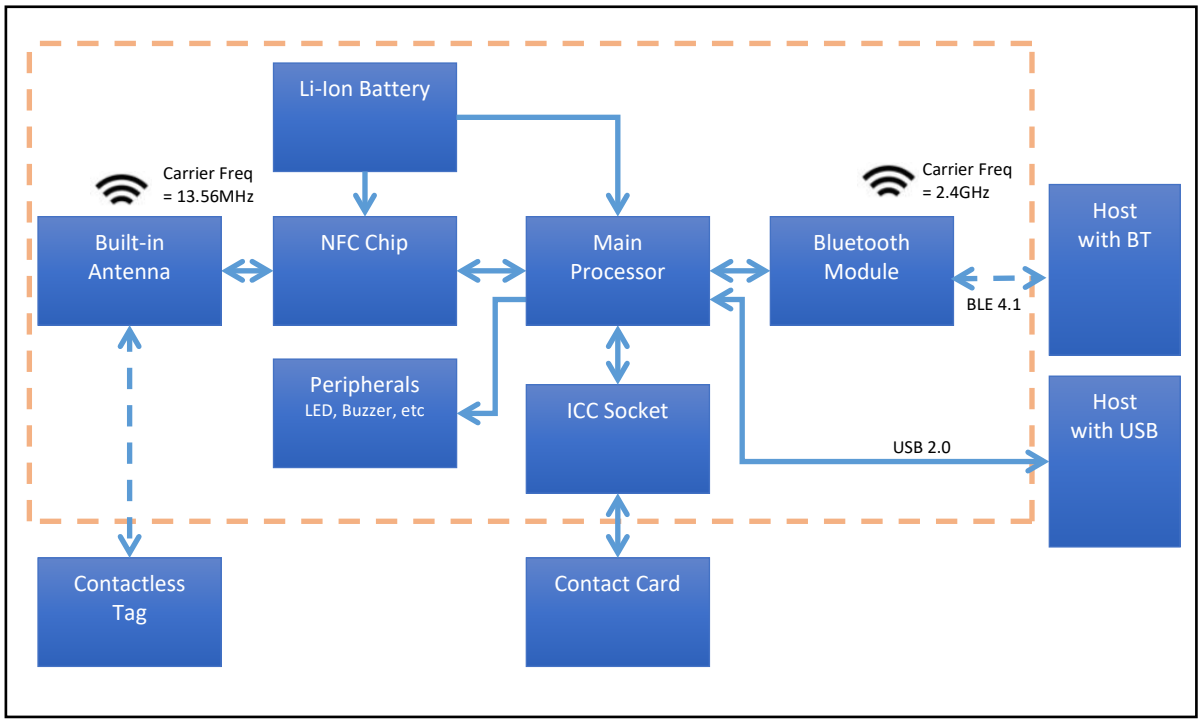


Figure 1: Hardware Architecture

3.1.1. Battery

The AMR220-C1 uses a rechargeable Lithium-ion battery, which has a capacity of 450 mAh.

3.1.1.1. Battery Charging

The battery of the AMR220-C1 may be charged by connecting it to a power outlet.

3.1.1.2. Battery Life

The battery life is dependent on the usage of the device. Below is an estimate of the battery life depending on various work conditions:

Mode	Estimated Battery Life
Working Mode: Contact Interface	9 days*(1)
Working Mode: Contactless Interface	7 days*(1)
OFF Mode	2 years

Table 2: Estimated Battery Lifespan

***Note:** Results may vary as it depends on the smart card used.*

(1) In Bluetooth mode, 10 operations per day with 1 minute operation run

3.1.2. User Interface

3.1.2.1. LED Behaviors

Charging LED

LED Status		Description
Charging LED On		Charging
Charging LED Off		Fully Charged

Bluetooth LED

LED Status	Description
Slow Blinking (0.5 sec - ON, 4.5 sec - OFF)	Paired
Slow Blinking (0.5 sec - ON, 1.5 sec - OFF)	Pairing
OFF	Bluetooth is Inactive

LED Status	
BT LED ON	
BT LED Off	

EMV Contactless LEDs Behaviors

***Note**: For more information, refer to EMV Contactless Specification: https://www.emvco.com/wp-content/uploads/2017/05/Book_A_Architecture_and_General_Rqmts_v2_6_Final_20160422011856105.pdf.*

3.1.2.2. Switch Behaviors

3.1.2.2.1. Power Switch Behaviors

- To turn on the device, press and hold the power switch for about 1-2 seconds.
- To turn off the device, press and hold the power switch until a beep sound is heard, then release the switch.

3.1.2.2.2. Bluetooth Switch Behaviors

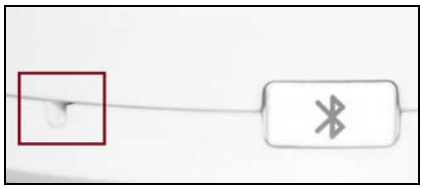
To activate/deactivate Bluetooth paring:

- To activate Bluetooth pairing, press the Bluetooth switch once.
- To deactivate Bluetooth pairing, press the Bluetooth switch twice within 2 seconds.

To set device into Firmware Update Mode:

- If the device is turned on, press and hold the Bluetooth switch for about 10 seconds.
- If the device is turned off, press the Bluetooth switch together with the power switch.

3.1.2.3. Device Reset Pinhole



To reset the device:

- To reset the device, locate the pinhole beside the Bluetooth switch, then press the reset button inside the pinhole using a pin.

3.2. Software Design

3.2.1. USB Interface

By using an MS-CCID driver, only a single slot device may be supported, and only the PICC interface may be used. In order to use both interfaces, the ACS driver is needed. The AMR220-C1 USB interface follows CCID protocol with two defined slots; one for ICC interface and one for PICC interface.

Since the AMR220-C1 is a CCID device, the host application is fully compatible with the PCSC standard.

***Note**: For more details, please refer to Microsoft MSDN Library or PCSC workgroup.*

Some usually used PCSC API will be described in **PCSC API**.

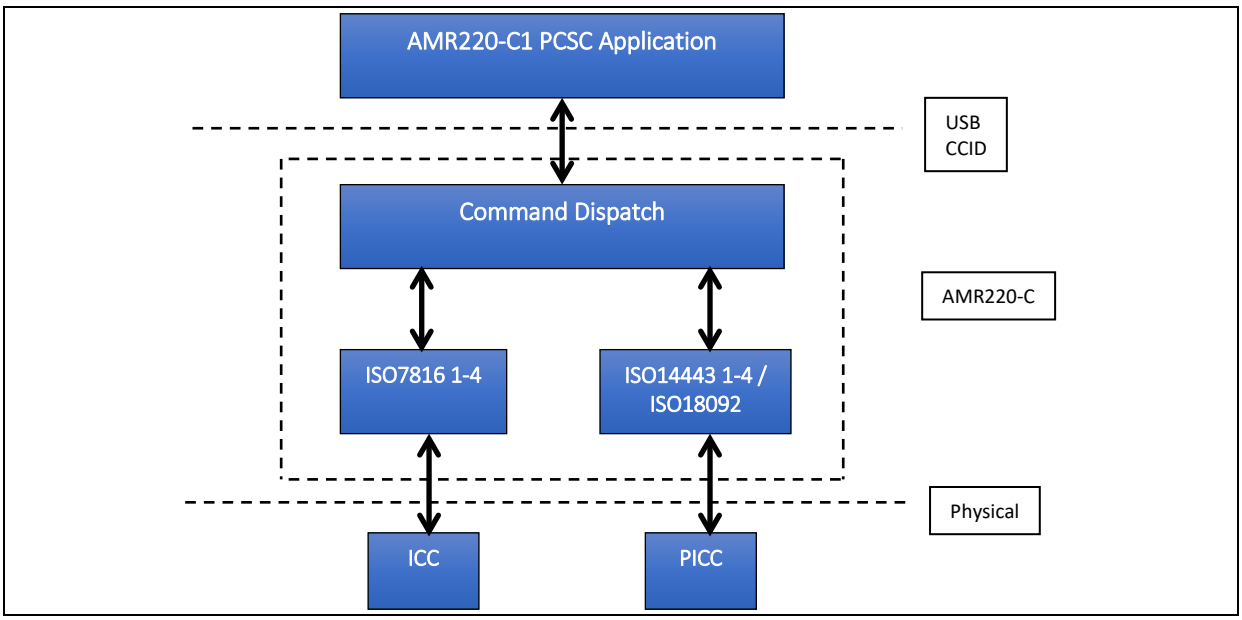


Figure 2: Software Architecture – USB Interface

3.2.2. Bluetooth Interface

The Bluetooth interface of the AMR220-C1 follows the BLE 4.1 standard. ACS offers a high level Android/IOS library to simplify application programming. The BT API is described in **Bluetooth Library**.

Below is the BLE architecture for reference.

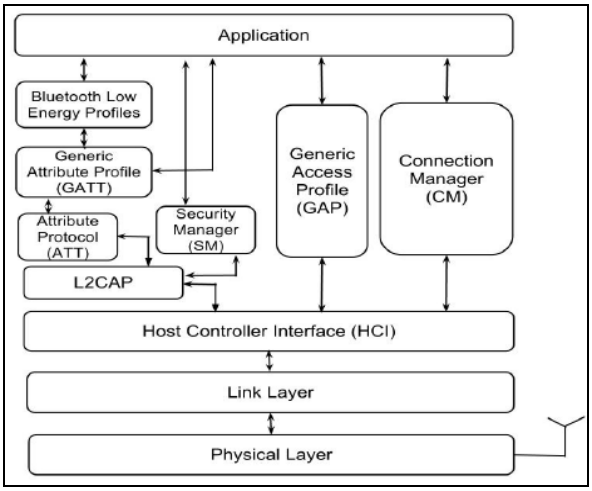


Figure 3: Software Architecture – Bluetooth Interface

4.0. Host Programming

4.1. PCSC API

This section describes some of the PCSC API for application programming using USB interface. For more details, please refer to Microsoft MSDN Library or PCSC workgroup.

4.1.1. SCardEstablishContext

The SCardEstablishContext function establishes the resource manager context within which database operations are performed.

Refer to: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379479%28v=vs.85%29.aspx

4.1.2. SCardListReaders

The SCardListReaders function provides the list of readers within a set of named reader groups, eliminating duplicates.

The caller supplies a list of reader groups, and receives the list of readers within the named groups. Unrecognized group names are ignored. This function only returns readers within the named groups that are currently attached to the system and available for use.

Refer to: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379793%28v=vs.85%29.aspx

4.1.3. SCardConnect

The SCardConnect function establishes a connection (using a specific resource manager context) between the calling application and a smart card contained by a specific reader. If no card exists in the specified reader, an error is returned.

Refer to: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379473%28v=vs.85%29.aspx

4.1.4. SCardControl

The SCardControl function gives you direct control of the reader. You can call it any time after a successful call to SCardConnect and before a successful call to SCardDisconnect. The effect on the state of the reader depends on the control code.

Refer to: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379474%28v=vs.85%29.aspx

Note: *Commands from* ***Escape Command*** *use this API for sending.*

4.1.5. ScardTransmit

The SCardTransmit function sends a service request to the smart card and expects to receive data back from the card.

Refer to: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379804%28v=vs.85%29.aspx

Note: *APDU Commands (i.e. the commands sent to connected card in* ***Pseudo APDUs for Contactless Interface****) use this API for sending.*

4.1.6. ScardDisconnect

The SCardDisconnect function terminates a connection previously opened between the calling application and a smart card in the target reader.

Refer to: http://msdn.microsoft.com/en-us/library/windows/desktop/aa379475%28v=vs.85%29.aspx

4.1.7. APDU Flow

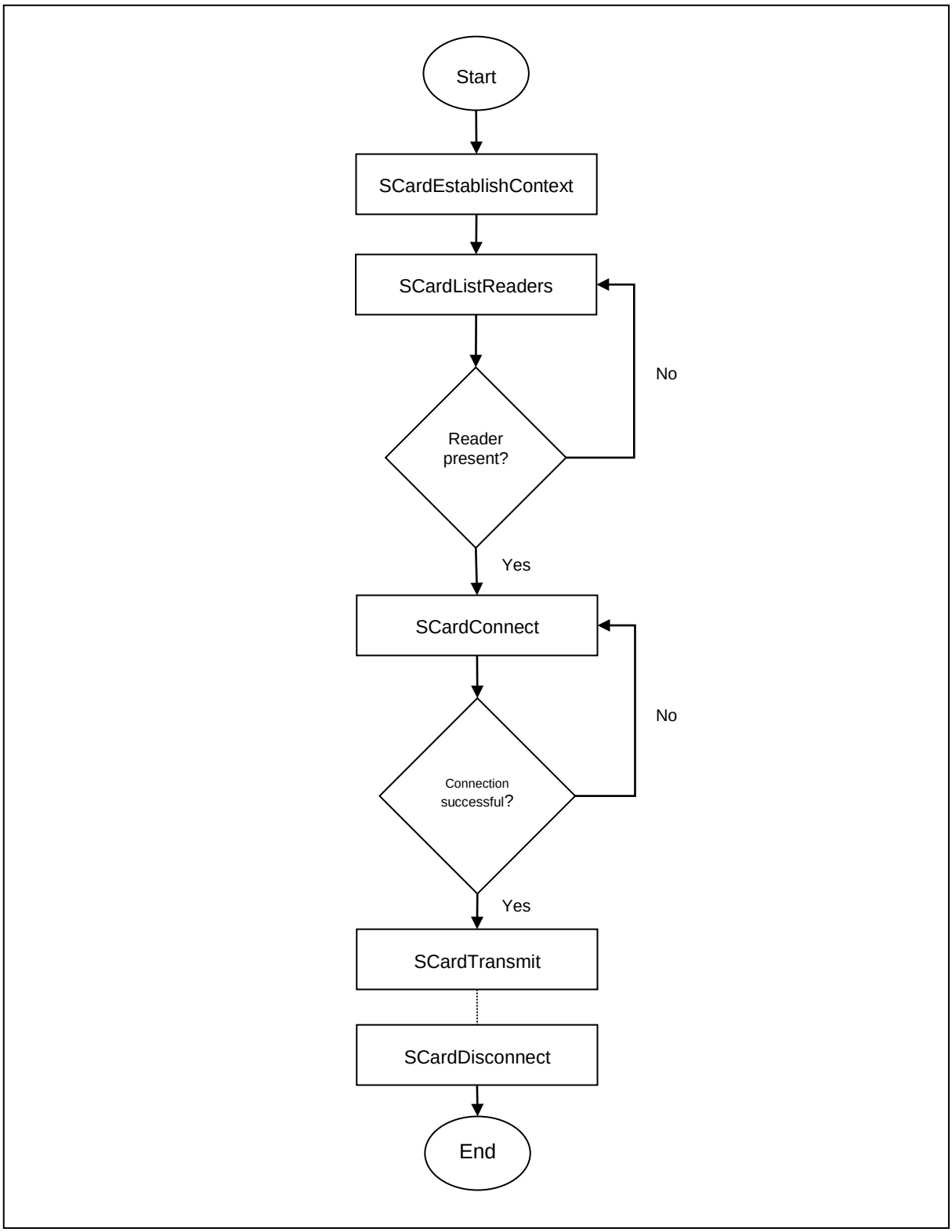


Figure 4: APDU Flow

4.1.8. Escape Command Flow

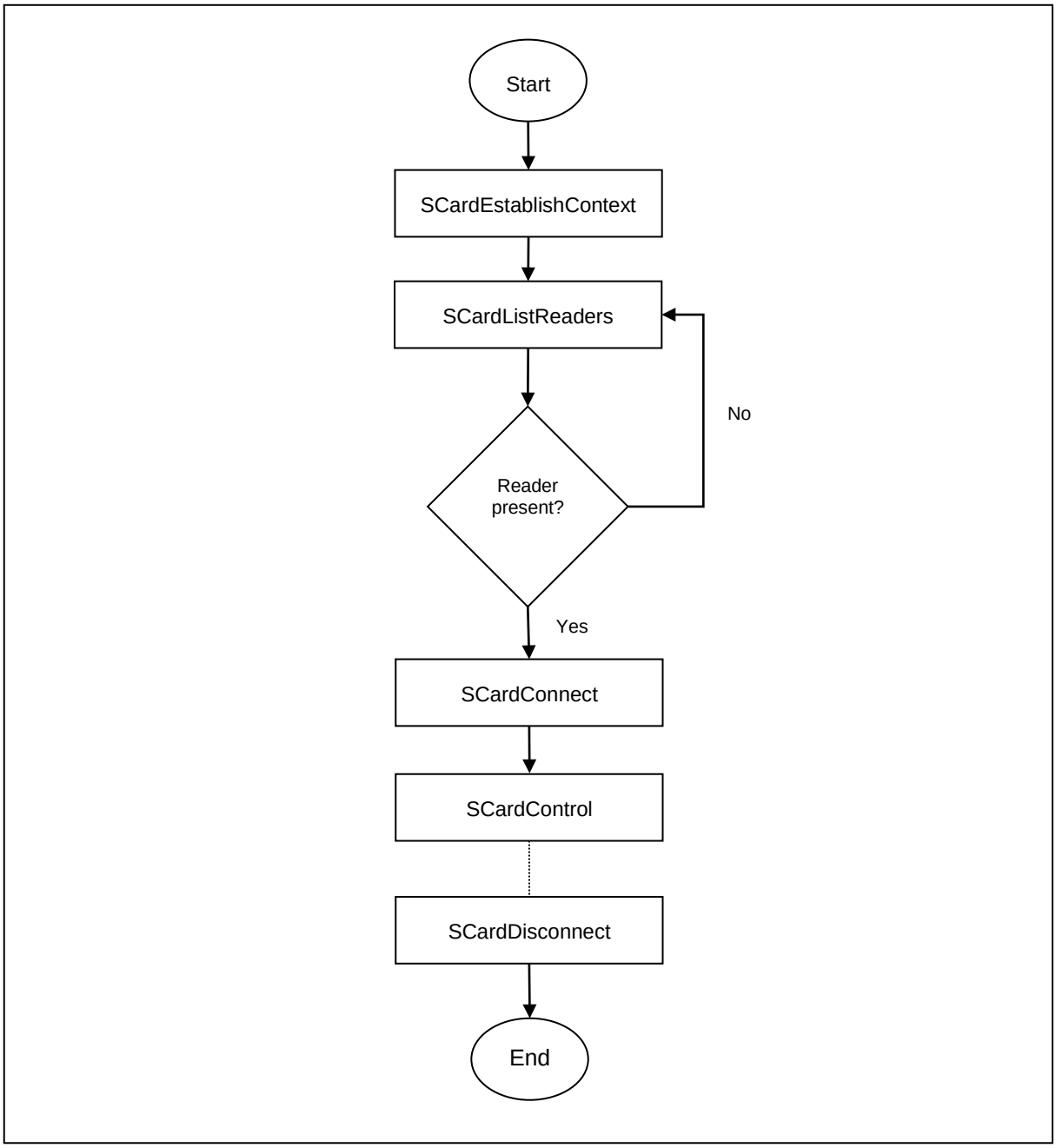


Figure 5: Escape Command Flow

4.2. Bluetooth Library

This section describes the ACS BT library for developer use. For more details, please refer to the Library package documents.

4.2.1. Setting Up BLE

If BLE is supported on the device, enable Bluetooth in order to connect card terminals. To enable Bluetooth, get the instance of BluetoothAdapter from BluetoothManager.

```
private BluetoothAdapter mBluetoothAdapter;
 ...
 // Initializes Bluetooth adapter.
final BluetoothManager bluetoothManager = (BluetoothManager)
      getSystemService(Context.BLUETOOTH_SERVICE);
mBluetoothAdapter = bluetoothManager.getAdapter();
```

To check whether Bluetooth is enabled or not, call isEnabled() from BluetoothAdapter. Then, call startActivityForResult() to request user permission to enable Bluetooth. The result will be returned from onActivityResult() implementation.

```
// Ensures Bluetooth is available on the device and it is enabled. If not,
// displays a dialog requesting user permission to enable Bluetooth.
if (mBluetoothAdapter == null || !mBluetoothAdapter.isEnabled()) {
   Intent enableBtIntent = new
   Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
 }
```

4.2.2. Initializing Java Smart Card I/O API

To use Java Smart Card I/O API with BLE card terminals, call getInstance() from BluetoothSmartCard. This class is a singleton and takes a Context as a parameter. If this method is called within Activity, they can pass this to the parameter.

```
private BluetoothTerminalManager mManager;
private TerminalFactory mFactory;
 ...
 // Get the Bluetooth terminal manager.
 mManager = BluetoothSmartCard.getInstance(this).getManager();

 if (mManager == null) {
     Toast.makeText(this, R.string.error_bluetooth_not_supported,
              Toast.LENGTH_SHORT).show();
     finish();
     return;
 }

 // Get the terminal factory.
 mFactory = BluetoothSmartCard.getInstance(this).getFactory();

 if (mFactory == null) {
     Toast.makeText(this, R.string.error_bluetooth_provider_not_found,
           Toast.LENGTH_SHORT).show();
     finish();
     return;
 }
```

4.2.3. Finding BLE Card Terminals

To find BLE card terminals, use startScan() method from BluetoothTerminalManager. Supply a terminal type and a callback for returning CardTerminal object.

Once the card terminal is found, stopScan() must be called to stop the scanning. This is to avoid the smart device battery from draining quickly, which will affect the operation.

For Android 6.0 and later, developers must request either ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION permission at run-time in order to scan BLE devices.

```
private Handler mHandler;
private Button mScanButton;
private TerminalAdapter mTerminalAdapter;
...
// Initialize Scan button.
mHandler = new Handler();
mScanButton = (Button) findViewById(R.id.activity_main_button_scan);
mScanButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // Request access coarse location permission.
        if (ContextCompat.checkSelfPermission(MainActivity.this,
              Manifest.permission.ACCESS_COARSE_LOCATION)
              != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(MainActivity.this,
               new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
               REQUEST_ACCESS_COARSE_LOCATION);
        } else {
           mScanButton.setEnabled(false);
           mTerminalAdapter.clear();

        // Start the scan.

        mManager.startScan(BluetoothTerminalManager.TERMINAL_TYPE_AMR220_C,
            new BluetoothTerminalManager.TerminalScanCallback() {
                @Override
                public void onScan(final CardTerminal terminal) {
                 runOnUiThread(new Runnable() {
                   @Override
                   public void run() {
                     mTerminalAdapter.addTerminal(terminal);
                  }
                });
              }
           });

         // Stop the scan.
         mHandler.postDelayed(new Runnable() {
          @Override
          public void run() {
          mManager.stopScan();
          mScanButton.setEnabled(true);
         }
        }, SCAN_PERIOD);
      }
    }
});
```

The result will be returned from onRequestPermissionsResult(). After the permission is granted, the application can start the scan.

```
@Override
public void onRequestPermissionsResult(int requestCode,
      @NonNull String[] permissions,
      @NonNull int[] grantResults) {
  if (requestCode == REQUEST_ACCESS_COARSE_LOCATION) {
      if ((grantResults.length > 0)
            && (grantResults[0] == PackageManager.PERMISSION_GRANTED)) {
          mScanButton.setEnabled(false);
          mTerminalAdapter.clear();

        // Start the scan.
         mManager.startScan(BluetoothTerminalManager.TERMINAL_TYPE_AMR220_C,
           new BluetoothTerminalManager.TerminalScanCallback() {
             @Override
             public void onScan(final CardTerminal terminal) {
               runOnUiThread(new Runnable() {
                 @Override
                 public void run() {
                   mTerminalAdapter.addTerminal(terminal);
                 }
               });
             }
           });

        // Stop the scan.
        mHandler.postDelayed(new Runnable() {
          @Override
          public void run() {
            mManager.stopScan();
            mScanButton.setEnabled(true);
          }
        }, SCAN_PERIOD);
      }
   } else {
      super.onRequestPermissionsResult(requestCode, permissions,
           grantResults);
   }
}
```

4.2.4. Connecting to a Card

To connect a card, call connect() from CardTerminal object to return a Card object. The available protocols are T=0, T=1, T=0 or T=1 and direct.

```
// Protocol:
// "T=0"    - T=0
// "T=1"    - T=1
// "*"      - T=0 or T=1
// "direct" - Direct mode
try {
    Card card = terminal.connect("*");
} catch (CardException e) {
    e.printStackTrace();
}
```

4.2.5. Disconnecting from the Card

After using the card, disconnect it by calling disconnect() from the Card object. To reset the card after disconnecting, pass true to the parameter.

```
try {
     card.disconnect(false);
 } catch (CardException e) {
     e.printStackTrace();
 }
```

4.2.6. Transmitting APDUs

If the card is connected successfully, open a channel to transmit APDUs. Call either getBasicChannel() or openLogicalChannel() from Card object. After a CardChannel object is returned, transmit APDUs using transmit(CommandAPDU) or transmit(ByteBuffer, ByteBuffer) from the CardChannel object.

```
byte[] command = { 0x00, (byte) 0x84, 0x00, 0x00, 0x08 };

 try {
     Card card = terminal.connect("*");
     CardChannel channel = card.getBasicChannel();
     CommandAPDU commandAPDU = new CommandAPDU(command);
     ResponseAPDU responseAPDU = channel.transmit(commandAPDU);
 } catch (CardException e) {

     e.printStackTrace();
 }
```

4.2.7. Transmitting Control Commands

If the card is connected successfully, call transmitControlCommand() from the Card object to transmit control commands.

```
int controlCode = BluetoothTerminalManager.IOCTL_ESCAPE;
 try {
     Card card = terminal.connect("direct");
     card.transmitControlCommand(controlCode, command);
 } catch (CardException e) {
     e.printStackTrace();
 }
```

4.2.8. Disconnecting from BLE Card Terminal

The library connects to a BLE card terminal automatically when calling connect() from the CardTerminal object. To terminate the Bluetooth connection manually, call disconnect() from BluetoothTerminalManager object.

```
mManager.disconnect(terminal);
```

5.0. Command Set

5.1. API between Tablet and AMR220-C1

***Note**: This section is only for the developers who will not use the ACS BT Library for the development of their own application. If you are using the ACS BT Library, please ignore this Section.*

5.1.1. BT Communication Frame format

There is a defined frame format for communication between the AMR220-C1 and a tablet.

The format is defined below:

STX (0x02)	Data Len MSB	Data Len LSB	Sequence Number	Frame Type	Data Field (N Bytes)	CheckSum

Where:

STX Start of Text, must be equal to 0x02

Data Len MSB Length of the Data Field, MSB (1 Byte)

Data Len LSB Length of the Data Field, LSB (1 Byte)

Sequence Number Sequence number for BT Message, increases per message, bit 7 indicates whether it is chaining or not (“1” = chaining package, “0” = end package)

Frame Type Communicate Frame type

Data Frame without Encrypt = 0x00 Encrypted Data Frame = 0x01

ACK Frame = 0x02 NAK Frame = 0x03

Abort Frame = 0x04 INT Frame = 0x05

Inter character timeout Frame = 0xF1 Checksum Incorrect Frame = 0xF2

Data Len Error Frame = 0xF3

Data Field: Message Body (N Bytes), Data Encrypt part

Checksum: XOR {Data Len MSB, Data Len LSB, Sequence Number, Frame Type, Data Field}

Status Reply：

1. **Received, or Chaining Accepted - ACK**
   Reply format:

0x02	0x00	0x00	Seq	0x02	CS

   After the ACK Reply, Response Frame will be followed.

2. **Receiving failure - Negative Acknowledgment - NAK**

Reply format:

0x02	0x00	0x00	Seq	0x03	CS

3. **Inter character timeout - the timeout value between each byte (for example 5ms)**

Reply format:

0x02	0x00	0x00	Seq	0xF1	CS

4. **Checksum checking - Checksum is incorrect**
   Reply format:

0x02	0x00	0x00	Seq	0xF2	CS

5. **Data Length Error - Data length is over the maximum limit**

Reply format:

0x02	0x00	0x00	Seq	0xF3	CS

5.1.2. Data Field Format – Command

This section defines the data field content format on the command frame.

Data Field				
INS (1 Byte)	Counter (1 Byte)	Command Payload Length (2 bytes)	Command Payload (N Bytes)	CS (1 Byte)

Where:

INS	Command Header (1 byte) The command header for specific the command feature
Counter	Counter for protection against replay attack (1 byte) Increment per command, for protection against replay attack
Command Payload Length	Length of the Command Payload (2 bytes)
Command Payload	Message on the command (N bytes)
CS	Checksum, XOR {INS, Counter, Command Payload Length, Command Payload}

5.1.3. Data Field Format – Response

This section defines the data field content format on the response frame.

Data Field				
RSP (1 Byte)	Counter (1 Byte)	Response Payload Length (2 bytes)	Response Payload (N Bytes)	CS (1 Byte)

Where:

RSP	Response Header (1 byte) The response header for specific the command header
Counter	Counter for protection against replay attack (1 byte) Increment per Response
Response Payload Length	Length of the Response Payload (2 bytes)
Response Payload	Message on the Response (N bytes)
CS	Checksum, XOR {INS, Counter, Response Payload Length, Response Payload}

5.1.4. BT Commands and Responses

5.1.4.1. Commands (From Smart Device to AMR220-C1)

Command Name	Description	INS
SPH_to_RDR_EmvExchangeData	Exchanges data between smart device and EMV L2 kernel	0x44
SPH_to_RDR_PcdPowerOn	Asks AMR220-C1 to poll PCD tag within the period defined	0x80
SPH_to_RDR_PcdPowerOff	Asks AMR220-C1 to deselect the activated PCD tag	0x81
SPH_to_RDR_PcdExAPDU	Asks AMR220-C1 to exchange APDU with the activated PCD tag ***Note**: SPH_to_RDR_PcdPowerOn must be completed first.*	0x82
SPH_to_RDR_IccPowerOn	Asks AMR220-C1 to activate ICC/SAM card	0xA0
SPH_to_RDR_IccPowerOff	Asks AMR220-C1 to power off the activated ICC/SAM card	0xA1
SPH_to_RDR_IccExAPDU	Asks AMR220-C1 to exchange APDU with the activated ICC/SAM card ***Note**: SPH_to_RDR_IccPowerOn must be completed first.*	0xA2
SPH_to_RDR_IccSetParameter	Asks AMR220-C1 to do PPS with the activated ICC/SAM card ***Note**: SPH_to_RDR_IccPowerOn must be completed first.*	0xA3
SPH_to_RDR_ExEscape	Asks AMR220-C1 to exchange Escape Command to control/configure peripherals	0xC0

Table 3: BT Commands From Smart Device to AMR220-C1

5.1.4.1.1. SPH_to_RDR_PcdPowerOn

This command asks the AMR220-C1 to poll PCD tag within the period defined.

Offset	Field	Size	Value	Description
0	INS	1	0x80	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	0x0006	Length of the Command Payload
4	Command Payload	6		Byte 1 – Card Type Bit 0 = 0 – Disable Type A Polling 1 – Enable Type A polling Bit 1 = 0 – Disable Type B Polling 1 – Enable Type B polling Bit 2 = 0 – Disable FeliCa212 Polling 1 – Enable FeliCa212 polling Bit 3 = 0 – Disable FeliCa424 Polling 1 – Enable FeliCa424 polling Bit 4-6 = RFU Bit 7 = 0 – No RATS send 1 – RATS Send Automatically Byte 2 – Polling Retry Byte 3 – 6 – Polling interval Interval between each polling (unit: 1ms)
10	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_PcdPowerOnRsp

5.1.4.1.2. SPH_to_RDR_PcdPowerOff

This command asks the AMR220-C1 to deselect the activated PCD tag.

Offset	Field	Size	Value	Description
0	INS	1	0x81	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	0x0000	Length of the Command Payload
4	Command Payload	0		-
4	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_PcdPowerOffRsp

5.1.4.1.3. SPH_to_RDR_PcdExAPDU

This command asks the AMR220-C1 to exchange APDU with the activated PCD tag.

***Note**: SPH_to_RDR_PcdPowerOn must be completed first.*

Offset	Field	Size	Value	Description
0	INS	1	0x82	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	N	Length of the Command Payload
4	Command Payload	N		APDU to activated PCD tag
N+4	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_PcdExAPDURsp

5.1.4.1.4. SPH_to_RDR_IccPowerOn

This command asks the AMR220-C1 to activate the ICC/SAM card.

Offset	Field	Size	Value	Description
0	INS	1	0xA0	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	0x0001	Length of the Command Payload
4	Command Payload	1		ICC Slot select 0x01 = ICC Slot
5	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_IccPowerOnRsp

5.1.4.1.5. SPH_to_RDR_IccPowerOff

This command asks the AMR220-C1 to power off the activated ICC/SAM card.

Offset	Field	Size	Value	Description
0	INS	1	0xA1	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	0x0001	Length of the Command Payload
4	Command Payload	1		ICC Slot select 0x01 = ICC Slot
5	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_IccPowerOffRsp

5.1.4.1.6. SPH_to_RDR_IccExAPDU

This command asks the AMR220-C1 to exchange APDU with the activated ICC/SAM card.

***Note**: SPH_to_RDR_IccPowerOn must be completed first.*

Offset	Field	Size	Value	Description
0	INS	1	0xA2	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	N	Length of the Command Payload
4	Command Payload	N		Byte 1 - ICC Slot select 0x01 = ICC Slot Byte 2 - … - APDU APDU to activated ICC card
N+4	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_IccExAPDURsp

5.1.4.1.7. SPH_to_RDR_IccSetParameter

This command asks the AMR220-C1 to do PPS with the activated ICC/SAM card.

***Note**: SPH_to_RDR_IccPowerOn must be completed first.*

Offset	Field	Size	Value	Description
0	INS	1	0xA3	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	5	Length of the Command Payload
4	Command Payload	5		Byte 1 - ICC Slot select 0x01 = ICC Slot Byte 2 - 5 - PPS
9	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_IccSetParameterRsp.

5.1.4.1.8. SPH_to_RDR_ExEscape

This command asks the AMR220-C1 to exchange Escape command to control/configure the peripherals.

Offset	Field	Size	Value	Description
0	INS	1	0xC0	Command Header
1	Counter	1		Command Counter
2	Command Payload Length	2	5	Length of the Command Payload
4	Command Payload	N		Escape command
N+4	CS	1		XOR for the above field

The response to this message is RDR_to_SPH_ExEscapeRsp

5.1.4.2. Responses (From AMR220-C1 to Smart Device)

Response Name	Description	INS
RDR_to_SPH_EmvExchangeData	Response for SPH_to_RDR_EmvExchangeData	0x54
RDR_to_SPH_PcdPowerOnRsp	Response for SPH_to_RDR_PcdPowerOn	0x90
RDR_to_SPH_PcdPowerOffRsp	Response for SPH_to_RDR_PcdPowerOff	0x91
RDR_to_SPH_PcdExAPDURsp	Response for SPH_to_RDR_PcdExAPDU	0x92
RDR_to_SPH_IccPowerOnRsp	Response for SPH_to_RDR_IccPowerOn	0xB0
RDR_to_SPH_IccPowerOffRsp	Response for SPH_to_RDR_IccPowerOff	0xB1
RDR_to_SPH_IccExAPDURsp	Response for SPH_to_RDR_IccExAPDU	0xB2
RDR_to_SPH_IccSetParameterRsp	Response for SPH_to_RDR_IccSetParameter	0xB3
RDR_to_SPH_ExEscapeRsp	Response for SPH_to_RDR_ExEscape	0xD0

Table 4: BT Responses From AMR220-C1 to Smart Device

5.1.4.2.1. RDR_to_SPH_PcdPowerOnRsp

This command is a response to SPH_to_RDR_PcdPowerOn.

Offset	Field	Size	Value	Description
0	RSP	1	0x90	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	N	Length of the Response Payload
4	Response Payload	N		Byte 1 – Error Code 0x00 = No Error Other = ref to Error Code table Byte 2 - … - Card ATR
N+4	CS	1		XOR for the above field

5.1.4.2.2. RDR_to_SPH_PcdPowerOffRsp

This command is a response to SPH_to_RDR_PcdPowerOff.

Offset	Field	Size	Value	Description
0	RSP	1	0x91	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	0x0001	Length of the Response Payload
4	Response Payload	1		0x00 = No Error Other = ref to Error Code table
5	CS	1		XOR for the above field

5.1.4.2.3. RDR_to_SPH_PcdExAPDURsp

This command is a response to RDR_to_SPH_PcdExAPDURsp.

Offset	Field	Size	Value	Description
0	RSP	1	0x92	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	N	Length of the Response Payload
4	Response Payload	N		Byte 1 – Error Code 0x00 = No Error Other = ref to Error Code table Byte 2 - … - APDU Response
N+4	CS	1		XOR for the above field

5.1.4.2.4. RDR_to_SPH_IccPowerOnRsp

This code is a response to SPH_to_RDR_IccPowerOn.

Offset	Field	Size	Value	Description
0	RSP	1	0xB0	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	N	Length of the Response Payload
4	Response Payload	N		Byte 1 - ICC Slot select 0x01 = ICC Slot Byte 2 – Error Code 0x00 = No Error Other = ref to Error Code table Byte 3 - … - Card ATR
N+4	CS	1		XOR for the above field

5.1.4.2.5. RDR_to_SPH_IccPowerOffRsp

This command is a response to SPH_to_RDR_IccPowerOff.

Offset	Field	Size	Value	Description
0	RSP	1	0xB1	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	0x0002	Length of the Response Payload
4	Response Payload	2		Byte 1 - ICC Slot select 0x01 = ICC Slot Byte 2 – Error Code 0x00 = No Error Other = ref to Error Code Table
6	CS	1		XOR for the above field

5.1.4.2.6. RDR_to_SPH_IccExAPDURsp

This command is a response to SPH_to_RDR_IccExAPDU.

Offset	Field	Size	Value	Description
0	RSP	1	0xB2	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	N	Length of the Response Payload
4	Response Payload	N		Byte 1 - ICC Slot select 0x01 = ICC Slot Byte 2 – Error Code 0x00 = No Error Other = ref to Error Code table Byte 3 - … - APDU Response
N+4	CS	1		XOR for the above field

5.1.4.2.7. RDR_to_SPH_IccSetParameterRsp

This command is a response to SPH_to_RDR_IccSetParameter.

Offset	Field	Size	Value	Description
0	RSP	1	0xB3	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	N	Length of the Response Payload
4	Response Payload	N		Byte 1 - ICC Slot select 0x01 = ICC Slot Byte 2 – Error Code 0x00 = No Error Other = ref to Error Code table Byte 3 - … - PPS Response
N+4	CS	1		XOR for the above field

5.1.4.2.8. RDR_to_SPH_ExEscapeRsp

This command is a response to SPH_to_RDR_ExEscape.

Offset	Field	Size	Value	Description
0	RSP	1	0xD0	Response Header
1	Counter	1		Response Counter
2	Response Payload Length	2	N	Length of the Response Payload
4	Response Payload	N		Escape Command Response
N+4	CS	1		XOR for the above field

5.2. Contactless Smart Card Protocol

5.2.1. ATR Generation

The ATR for PICC interface follows PCSC Specification.

Note: *For more information, refer to http://pcscworkgroup.com/Download/Specifications/pcsc3_v2.01.09.pdf.*

5.2.1.1. ATR format for ISO 14443 Part 3 PICCs.

Byte	Value (Hex)	Designation	Description
0	0x3B	Initial	
1	0x8N	T0	Higher Nibble "8" means: • TA1, TB1, and TC1 Absent, TD1 available Lower Nibble "N" means: • the number of historical bytes
2	0x80	TD1	Higher Nibble "8" means: • TA2, TB2 and TC2 Absent, TD2 available Lower nibble "0" means: • Support Protocol T = 0
3	0x01	TD2	Higher nibble "0" means: • TA3, TB3, TC3 and TD3 Absent Lower nibble "1" means: • Support Protocol T = 1
4 To 3+N	0x80	T1	Category indicator byte "80 means" A status indicator may be present in an optional COMPACT-TLV data object
	0x4F	… Tk	Application identifier Presence Indicator
	0x0C		Length
	RID		Registered Application Provider Identifier (RID) Equal to "0xA0 0x00 0x00 0x03 0x06"
	SS		Byte for Standard
	C0 .. C1		Bytes for Card Name
	0x00 0x00 0x00 0x00	RFU	RFU
4+N	UU	TCK	Exclusive-or of all the bytes T0 to Tk

E.g. ATR for Mifare 1K = {0x3B 0x8F 0x80 0x01 0x80 0x4F 0x0C 0xA0 0x00 0x00 0x03 0x06 0x03 0x00 0x01 0x00 0x00 0x00 0x00 0x6A}

Length (YY) = 0x0C

RID = {0xA0 0x00 0x00 0x03 0x06} (PC/SC Workgroup)

Standard (SS) = 0x03 (ISO14443A, Part 3)

Card Name (C0 .. C1) = {0x00 0x01} (Mifare 1K)

Standard (SS)

0x03: ISO14443A, Part 3 0x11: FeliCa

Card Name (C0 .. C1)

0x00 0x01: Mifare 1K
0x00 0x02: Mifare 4K
0x00 0x03: Mifare Ultralight
0x00 0x26: Mifare Mini
0x00 0x30: Topaz
0x00 0x3B: FeliCa
0x00 0x38: Mifare Plus SL2_2K
0x00 0x39: Mifare Plus SL2_4K
0xFF [SAK]: undefined tags

5.2.1.2. ATR format for ISO 14443 Part 4 PICCs.

<table>
<tr><th>Byte</th><th>Value (Hex)</th><th>Designation</th><th>Description</th></tr>
<tr><td>0</td><td>0x3B</td><td>Initial</td><td></td></tr>
<tr><td>1</td><td>0x8N</td><td>T0</td><td>Higher Nibble “8” means:
• TA1, TB1 and TC1 Absent, TD1 available
Lower Nibble “N” means:
• the number of historical bytes</td></tr>
<tr><td>2</td><td>0x80</td><td>TD1</td><td>Higher Nibble “8” means:
• TA2, TB2 and TC2 Absent,TD2 available
Lower Nibble “0” means:
• Support Protocol T = 0</td></tr>
<tr><td>3</td><td>0x01</td><td>TD2</td><td>Higher Nibble “0” means:
• TA3, TB3, TC3, TD3 Absent
Lower Nibble “1” means:
• Support Protocol T = 1</td></tr>
<tr><td rowspan="2">4
to
3 + N</td><td>XX</td><td>T1</td><td rowspan="2">Historical Bytes:

ISO14443A:
The historical bytes from ATS response.
Refer to the ISO14443-4 specification.

ISO14443B:
<table><tr><th>Byte1-4</th><th>Byte5-7</th><th>Byte8</th></tr><tr><td>Application Data from ATQB</td><td>Protocol Info Byte from ATQB</td><td>Higher Nibble = MBLI from ATTRIB command
Lower Nibble (RFU) = 0</td></tr></table></td></tr>
<tr><td>XX
XX
XX

…
Tk</td><td></td></tr>
<tr><td>4+N</td><td>UU</td><td>TCK</td><td>Exclusive-oring of all the bytes T0 to Tk</td></tr>
</table>

E.g. 1. ATR for Desfire = {0x3B 0x81 0x80 0x01 0x80 0x80} // 6 bytes of ATR

***Note**: Use the APDU “0xFF 0xCA 0x01 0x00 0x00” to distinguish the ISO14443A-4 and ISO14443B-4 PICCs, and retrieve the full ATS if available. ISO14443A-3 and ISO14443B-3/4 PICCs do have ATS returned.*

APDU Command = 0xFF 0xCA 0x01 0x00 0x00

APDU Response = 0x06 0x75 0x77 0x81 0x02 0x80 0x90 0x00

ATS = {0x06 75 77 81 02 80}

E.g. 2. ATR for EZ-link = {0x3B 0x88 0x80 0x01 0x1C 0x2D 0x94 0x11 0xF7 0x71 0x85 0x00 0xBE}

Application Data of ATQB = 0x1C 0x2D 0x94 0x11

Protocol Information of ATQB = 0xF7 0x71 0x85

MBLI of ATTRIB = 0x00

5.2.2. Pseudo APDUs for Contactless Interface

5.2.2.1. Get Data Command

The Get Data command retrieves information about the inserted command depending on the inserted card. It can be used for different kinds of contactless cards.

Get Data Command Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Get Data	0xFF	0xCA	0x00 0x01	0x00	0x00

If P1 = 0x00, UID Response Format (UID + 2 bytes)

Response	Data Out					
Result	UID (LSB)	…	…	UID (MSB)	SW1	SW2

If P1 = 0x01, ATS of a ISO 14443 A card Response Format (ATS + 2 bytes)

Response	Data Out		
Result	ATS	SW1	SW2

Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Warning	0x62	0x82	End of UID/ATS reached before Le bytes (Le is greater than UID Length).
Error	0x6C	XX	Wrong length (wrong number Le: 'XX' encodes the exact number) if Le is less than the available UID length.
Error	0x63	0x00	The operation failed.
Error	0x6A	0x81	The function is not supported

5.2.2.2. PCSC 2.0 part 3 (version 2.02 or above) Related APDU

PCSC2.0 part 3 commands are used to transparently pass data from an application to a contactless tag, return the received data transparently to the application, and perform protocol switch simultaneously.

5.2.2.2.1. Command and Response APDU Format

Command Format:

CLA	INS	P1	P2	Lc	Data In
0xFF	0xC2	0x00	Function	DataLen	Data[DataLen]

Functions 1 Byte

0x00 = Manage Session

0x01 = Transparent Exchange

0x02 = Switch Protocol

Other = RFU

Response Format:

Data Out	SW1	SW2
Data Field BER-TLV encoded		

Every command returns SW1 and SW2 together with the response data field (if available). SW1 SW2 is according to ISO 7816. SW1 SW2 from the following C0 data object should also be used.

The C0 data element format:

Tag	Length (1btye)	SW2
0xC0	0x03	Error Status

Error Status Description

Error Status	Description
XX SW1 SW2	XX = number of the bad data object in the APDU 00 = general error of APDU 01 = error in the 1st data object 02 = error in the 2nd data object
0x00 0x90 0x00	No error occurred
XX 0x62 0x82	Data object XX warning, requested information not available
XX 0x63 0x00	No information
XX 0x63 0x01	Execution stopped due to failure in other data object
XX 0x6A 0x81	Data object XX not supported
XX 0x67 0x00	Data object XX with unexpected length

Error Status	Description
XX 0x6A 0x80	Data object XX with unexpected vale
XX 0x64 0x00	Data Object XX execution error (no response from IFD)
XX 0x64 0x01	Data Object XX execution error (no response from ICC)
XX 0x6F 0x00	Data object XX failed, no precise diagnosis

The first value byte indicates the number of the erroneous data object XX and the last two bytes indicate the explanation of the error. Further SW1 SW2 values, according to ISO 7816, are allowed.

If there are more than one data objects in the C-APDU field and one data object failed, IFD can process the following data objects if they do not depend on the failed data objects.

5.2.2.2.2. Manage Session Command

Manage Session Commands are used to manage the transparent session, including starting a transparent session, ending a transparent session, managing the operation environment, and managing the capabilities of the IFD within the transparent session.

Manage Session Command

Command	Class	INS	P1	P2	Lc	Data In
ManageSession	0xFF	0xC2	0x00	0x00	DataLen	DataObject (N bytes)

Where:

Data Object (1 Byte)

Tag	Data Object
0x80	Version data Object
0x81	Start Transparent Session
0x82	End Transparent Session
0x83	Turn Off RF Field
0x84	Turn On RF Field
0x5F46	Timer
0xFF6D	Get Parameter
0xFF6E	Set Parameter

Manage Session Response Data Object

Tag	Data Object
0xC0	Generic Error status
0x80	Version data object
0xFF6D	IFD parameter data object

5.2.2.2.2.1. Start Session Data Object

This command starts a transparent session. Auto-polling is disabled from the start to the end of the session.

Start Session Data Object

Tag	Length (1btye)	Value
0x81	0x00	-

5.2.2.2.2.2. End Session Data Object

This command ends the transparent session. Auto-polling will be reset.

End Session Data Object

Tag	Length (1btye)	Value
0x82	0x00	-

5.2.2.2.2.3. Version Data Object

This command returns the version number of the IFD handler.

Version Data Object

Tag	Length (1btye)	Value		
0x80	0x03	Major	Minor	Build

5.2.2.2.2.4. Turn Off the RF Data Object

This command turns off the antenna field.

Turn off the RF field Data Object

Tag	Length (1btye)	Value
0x83	0x00	-

5.2.2.2.2.5. Turn On the RF Data Object

This command turns on the antenna field.

Turn on the RF field Data Object

Tag	Length (1btye)	Value
0x84	0x00	-

5.2.2.2.2.6. Timer Data Object

This command creates a 32-bit timer data object (unit of 1us).

For example, if there is timer data object with 5000us between RF turn off data object and RF turn on data object, the reader will turn off the RF field at about 5000us and then turn it on again.

Timer Data Object

Tag	Length (1btye)	Value
0x5F46	0x04	Timer (4bytes)

5.2.2.2.2.7. Get Parameter Data Object

This command get different parameters from the IFD.

Get Parameter Data Object

Tag	Length (1btye)	Value		
		Tag	Len	Value
0xFF6D	Var	TLV_Objects		

TLV_Objects:

Parameters Requested	Tag	Length
Frame size for IFD integer (FSDI)	0x01	0x00
Frame size for ICC integer (FSCI)	0x02	0x00
Frame waiting time integer (FWTI)	0x03	0x00
Max. Communication Speed supported by the IFD	0x04	0x00
Communication Speed of the ICC	0x05	0x00
Modulation Index	0x06	0x00
PCB for ISO/IEC14443	0x07	0x00
CID for ISO/IEC14443	0x08	0x00
NAD for ISO/IEC14443	0x09	0x00

Parameters Requested	Tag	Length
Param 1 – 4 for for ISO/IEC14443 type B	0x0A	0x00

5.2.2.2.2.8. Set Parameter Data Object

Set different parameters from the IFD.

Set Parameter Data Object

Tag	Length (1btye)	Value		
		Tag	Len	Value
0xFF6E	Var	TLV_Objects		

TLV_Objects:

Parameters Requested	Tag	Length
Frame size for IFD integer (FSDI)	0x01	0x01
Frame size for ICC integer (FSCI)	0x02	0x01
Frame waiting time integer (FWTI)	0x03	0x01
Max. Communication Speed supported by the IFD	0x04	0x01
Communication Speed of the PICC	0x05	0x01
Modulation Index	0x06	0x01
PCB for ISO/IEC14443	0x07	0x01
CID for ISO/IEC14443	0x08	0x01
NAD for ISO/IEC14443	0x09	0x01
Param 1 – 4 for for ISO/IEC14443 type B	0x0A	0x04

5.2.2.2.3. Transparent Exchange Command

Transparent Exchange Commands are used to transmit and receive any bit or bytes from ICC.

Transparent Exchange Command

Command	Class	INS	P1	P2	Lc	Data In
TranspEx	0xFF	0xC2	0x00	0x01	DataLen	DataObject (N bytes)

Where:

Data Object (1 Byte)

Tag	Data Object
0x90	Transmission and Reception Flag
0x91	Transmission Bit Framing
0x92	Reception Bit Framing
0x93	Transmit
0x94	Receive
0x95	Transceive – Transmit and Receive
0xFF6D	Get Parameter
0xFF6E	Set Parameter

Transparent Exchange Response Data Object

Tag	Data Object
0xC0	Generic Error status
0x92	Number of valid bits in the last byte of received data
0x96	Response Status
0x97	ICC response
0xFF6D	IFD parameter data object

5.2.2.2.3.1. Transmission and Reception Flag Data Object

Defines the framing and RF parameters for the following transmission.

Transmission and Reception Flag Data Object

Tag	Length (1btye)	Value	
		bit	Description
0x90	0x02	0	0 – append CRC in the transmit data 1 – do not append CRC in the transmit data
		1	0 – discard CRC from the received data 1 – do not discard CRC from the received data (i.e. no CRC checking)
		2	0 – insert parity in the transmit data 1 – do not insert parity
		3	0 – expect parity in received date 1 – do not expect parity (i.e. no parity checking)

Tag	Length (1btye)	Value	
		bit	Description
		4	0 – append protocol prologue in the transmit data or discard from the response 1 – do not append or discard protocol prologue if any (e.g. PCB, CID, NAD)
		5-15	RFU

5.2.2.2.3.2. Transmission bit Framing Data Object

Defines the number of valid bits of the last byte of data in the transmit or transceive.

Transmission bit Framing Data Object

Tag	Length (1btye)	Value	
		bit	Description
0x91	0x01	0-2	Number of valid bits of the last byte (0 means all bits are valid)
		3-7	RFU

Transmission bit framing data object shall be together with “transmit” or “transceive” data object only.

If this data object is absent, it means all bits are valid.

5.2.2.2.3.3. Reception bit Framing Data Object

For the command APDU, this data object is used to define the number of expected valid bits of the last byte of data received.

For the response APDU, this data object is used to mention the number of valid bits in the last byte of received data.

Reception bit Framing Data Object

Tag	Length (1btye)	Value	
		bit	Description
0x92	0x01	0-2	Number of valid bits of the last byte (0 means all bits are valid)
		3-7	RFU

If this data object is absent, it means all bits are valid.

5.2.2.2.3.4. Transmit Data Object

To transmit the data from IFD to the ICC, no response is expected from the ICC after transmission is completed.

Transmit Data Object

Tag	Length (1btye)	Value
0x93	DataLen	Data (N bytes)

5.2.2.2.3.5. Receive Data Object

To force the reader into receiving mode within the specified time, which is given in the following timer object.

Receive Data Object

Tag	Length (1btye)	Value
0x94	0x00	-

5.2.2.2.3.6. Transceive Data Object

This is for transmitting and receiving data from the ICC. After transmission is complete, the reader will wait until the time which is given in timer data object.

If no timer data object is defined in the data field, then the reader will wait for the time given in the Set parameter FWTI data object.

If no FWTI is set, then the reader will wait about 302us

Transceive Data Object

Tag	Length (1btye)	Value
0x95	DataLen	Data (N Bytes)

5.2.2.2.3.7. Response Status Data Object

This notifies about the received data status inside the response.

Response Status Data Object

Tag	Length (1btye)	Value		
		Byte 0		Byte 1
		Bit	Description	
0x96	0x02	0	0 – CRC is OK or no checked 1 – CRC check fail	If collision is detected, these bytes will tell the collision position. Otherwise, “00h” will be shown
		1	0 – no collision 1 – collision detected	

Tag	Length (1btye)	Value		
		Byte 0		Byte 1
		Bit	Description	
		2	0 – no parity error 1 – parity error detected	
		3	0 – no framing error 1 – framing error detected	
		4 - 7	RFU	

5.2.2.2.3.8. Response Data Object

Notifies about the received data inside the response.

Response Data Object

Tag	Length (1btye)	Value
0x97	DataLen	ReplyData (N Byte)

5.2.2.2.4. Switch Protocol Command

Used for specific protocol and different layers of the standard within the transparent session.

Switch Protocol Command

Command	Class	INS	P1	P2	Lc	Data In
SwProtocol	0xFF	0xC2	0x00	0x02	DataLen	DataObject (N bytes)

Where:

Data Object (1 Byte)

Tag	Data Object
0x8F	Switch Protocol Data Object
0xFF6D	Get Parameter
0xFF6E	Set Parameter

Switch Protocol Response Data Object

Tag	Data Object
0xC0	Generic Error status
0xFF6D	IFD parameter data object

5.2.2.2.4.1. Switch Protocol Data Object

Used for specific protocol and different layers of the standard.

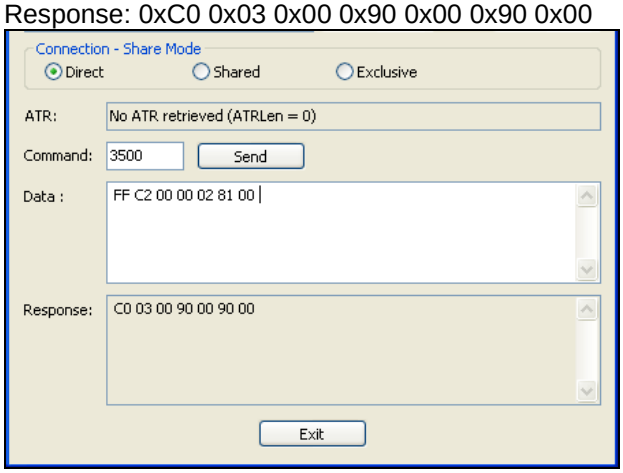
Switch Protocol Data Object

Tag	Length (1btye)	Value	
		Byte 0	Byte 1
0x8F	0x02	0x00 – ISO/IEC14443 Type A 0x01 – ISO/IEC14443 Type B 0x03 – FeliCa Other – RFU	0x00 – if no layer separation 0x02 – Switch to Layer 2 0x03 – Switch or activate to layer 3 0x04 – activate to layer 4 Other - RFU

5.2.2.2.5. PCSC 2.0 part 3 Example

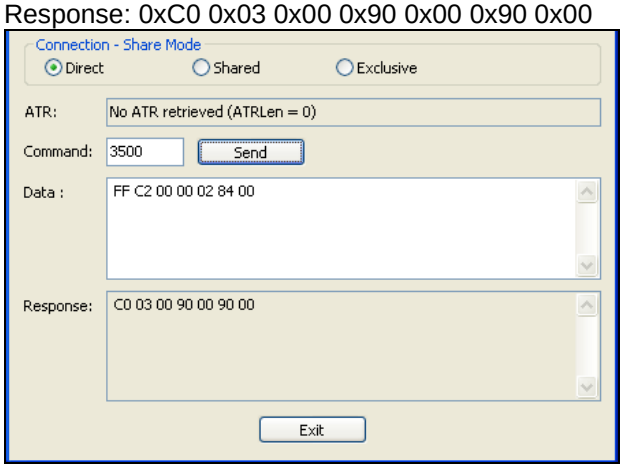
Step 1. Start Transparent Session

Command: 0xFF 0xC2 0x00 0x00 0x02 0x81 0x00
Response: 0xC0 0x03 0x00 0x90 0x00 0x90 0x00



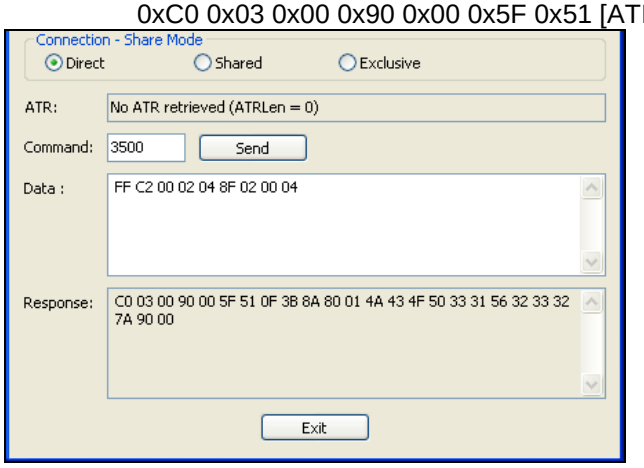
Step 2. Antenna Field On

Command: 0xFF 0xC2 0x00 0x00 0x02 0x84 0x00
Response: 0xC0 0x03 0x00 0x90 0x00 0x90 0x00



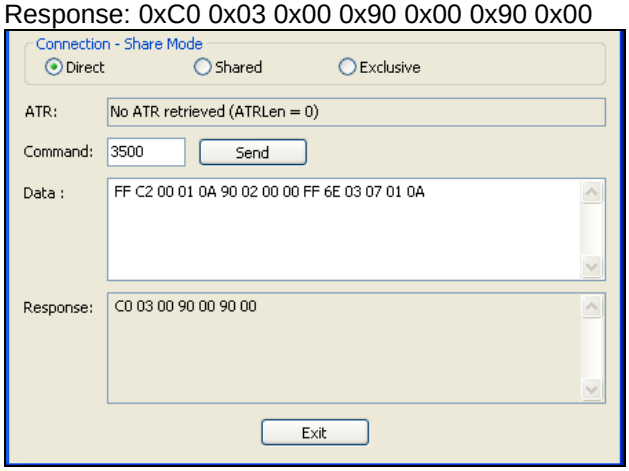
Step 3. ISO14443-4A Active

Command: 0xFF 0xC2 0x00 0x02 0x04 0x8F 0x02 0x00 0x04
Response: 0xC0 0x03 0x01 0x64 0x01 0x90 0x00 (if no card present)
0xC0 0x03 0x00 0x90 0x00 0x5F 0x51 [ATR] 0x90 0x00



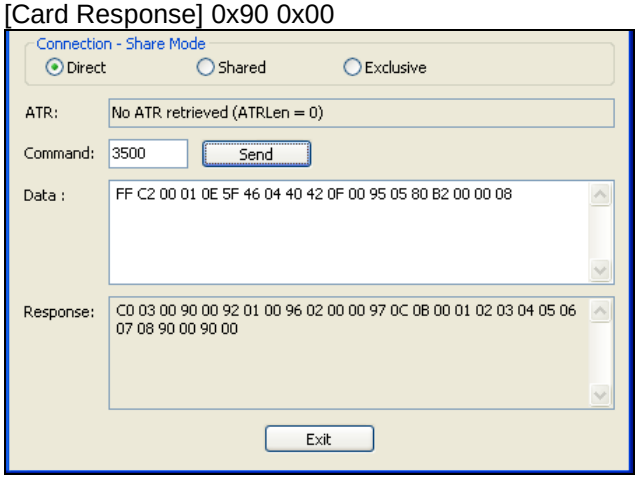
Step 4. Set the PCB to 0x0A and enable the CRC, parity and protocol prologue in the transmit data

Command: 0xFF 0xC2 0x00 0x01 0x0A 0x90 0x02 0x00 0x00 0xFF 0x6E 0x03 0x07 0x01 0x0A
Response: 0xC0 0x03 0x00 0x90 0x00 0x90 0x00



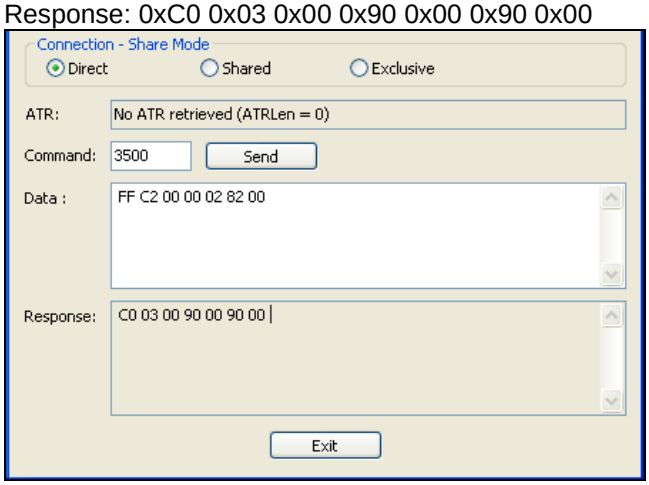
Step 5. Send the APDU "0x80 0xB2 0x00 0x00 0x08" to card and get response

Command: 0xFF 0xC2 0x00 0x01 0x0E 0x5F 0x46 0x04 0x40 0x42 0x0F 0x00 0x95 0x05 0x80 0xB2 0x00 0x00 0x08
Response: 0xC0 0x03 0x00 0x90 0x00 0x92 0x01 0x00 0x96 0x02 0x00 0x00 0x97 0x0C [Card Response] 0x90 0x00



Step 6. End Transparent Session

Command: 0xFF 0xC2 0x00 0x00 0x02 0x82 0x00
Response: 0xC0 0x03 0x00 0x90 0x00 0x90 0x00



5.2.2.3. PICC Commands (T=CL Emulation) for MIFARE 1K/4K MEMORY Cards

5.2.2.3.1. Load Authentication Keys

The "Load Authentication Keys" command loads the authentication keys into the reader. The authentication keys are used to authenticate the particular sector of the MIFARE 1K/4K Memory Card. Only volatile key location is available for AMR220-C1.

Load Authentication Keys APDU Format (11 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Load Authentication Keys	0xFF	0x82	Key Structure	Key Number	0x06	Key (6 bytes)

Key Structure (1 Byte):

0x00 = Key is loaded into the reader volatile memory.

Key Number (1 Byte):

0x00 ~ 0x01 = Volatile memory for storing a temporary key. The key disappears once the reader is disconnected from the PC. Two volatile keys are provided. The volatile key can be used as a session key for different sessions.
Default Value = {0xFF 0xFF 0xFF 0xFF 0xFF 0xFF}

Key (6 Bytes):

The key value loaded into the reader. E.g. {0xFF 0xFF 0xFF 0xFF 0xFF 0xFF}

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

Example:

// Load a key {0xFF 0xFF 0xFF 0xFF 0xFF FF} into the volatile memory location 0x00.

APDU = {0xFF 0x82 0x00 0x00 0x06 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF}

5.2.2.3.2. Authentication for MIFARE 1K/4K

The “Authentication command” uses the keys stored in the reader to do authentication with the MIFARE 1K/4K card. Two types of authentication keys are used: TYPE_A and TYPE_B.

Authentication APDU Format #1 (6 Bytes)

Command	Class	INS	P1	P2	P3	Data In
Authentication	0xFF	0x88	0x00	Block Number	Key Type	Key Number

Authentication APDU Format #2 (10 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Authentication	0xFF	0x86	0x00	0x00	0x05	Authenticate Data Bytes

Authenticate Data Bytes (5 Byte):

Byte1	Byte 2	Byte 3	Byte 4	Byte 5
Version 0x01	0x00	Block Number	Key Type	Key Number

Block Number (1 Byte):

The memory block to be authenticated

For MIFARE 1K Cards, it has a total of 16 sectors and each sector consists of 4 consecutive blocks. E.g.
Sector 0x00 consists of Blocks {0x00, 0x01, 0x02 and 0x03}
Sector 0x01 consists of Blocks {0x04, 0x05, 0x06 and 0x07}
the last sector 0x0F consists of Blocks {0x3C, 0x3D, 0x3E and 0x3F}

Once the authentication is done successfully, there is no need to do the authentication again, provided that the blocks to be accessed belong to the same sector. Please refer to the MIFARE 1K/4K specification for more details.

***Note**: Once the block is authenticated successfully, all the blocks belonging to the same sector are accessible.*

Key Type (1 Byte):

0x60 = Key is used as a TYPE A key for authentication.

0x61 = Key is used as a TYPE B key for authentication.

Key Number (1 Byte):

0x00 ~ 0x01 = Volatile memory for storing keys. The keys disappear when the reader is disconnected from the PC. Two volatile keys are provided. The volatile key can be used as a session key for different sessions.

Load Authentication Keys Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Load Authentication Keys Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

MIFARE 1K Memory Map

Sectors (Total 16 sectors) (Each sector consists of 4 consecutive blocks)	**Data Blocks** (3 blocks, 16 bytes per block)	**Trailer Block** (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
…	…	…
…	…	…
Sector 14	0x38 ~ 0x0A	0x3B
Sector 15	0x3C ~ 0x3E	0x3F

1K Bytes

MIFARE 4K Memory Map

Sectors (Total 32 sectors) (Each sector consists of 4 consecutive blocks)	**Data Blocks** (3 blocks, 16 bytes per block)	**Trailer Block** (1 block, 16 bytes)
Sector 0	0x00 ~ 0x02	0x03
Sector 1	0x04 ~ 0x06	0x07
…	…	…
…	…	…
Sector 30	0x78 ~ 0x7A	0x7B
Sector 31	0x7C ~ 0x7E	0x7F

2K Bytes

Sectors (Total 8 sectors) (Each sector consists of 16 consecutive blocks)	Data Blocks (15 blocks, 16 bytes per block)	Trailer Block (1 block, 16 bytes)
Sector 32	0x80 ~ 0x8E	0x8F
Sector 33	0x90 ~ 0x9E	0x9F
…	…	…
…	…	…
Sector 38	0xE0 ~ 0xEE	0xEF
Sector 39	0xF0 ~ 0xFE	0xFF

2K Bytes

Examples:

// to authenticate the Block 0x04 with a {TYPE A, key number 0x00}

// PC/SC V2.01, Obsolete

APDU = {0xFF 0x88 0x00 0x04 0x60 0x00};

// to authenticate the Block 0x04 with a {TYPE A, key number 0x00}

// PC/SC V2.07

APDU = {0xFF 0x86 0x00 0x00 0x05 0x01 0x00 0x04 0x60 0x00}

***Note**: MIFARE Ultralight does not need to do any authentication. The memory is free to access.*

MIFARE Ultralight Memory Map

Byte Number	0	1	2	3	Page
Serial Number	SN0	SN1	SN2	BCC0	0
Serial Number	SN3	SN4	SN5	SN6	1
Internal / Lock	BCC1	Internal	Lock0	Lock1	2
OTP	OPT0	OPT1	OTP2	OTP3	3
Data read/write	Data0	Data1	Data2	Data3	4
Data read/write	Data4	Data5	Data6	Data7	5
Data read/write	Data8	Data9	Data10	Data11	6
Data read/write	Data12	Data13	Data14	Data15	7
Data read/write	Data16	Data17	Data18	Data19	8
Data read/write	Data20	Data21	Data22	Data23	9
Data read/write	Data24	Data25	Data26	Data27	10
Data read/write	Data28	Data29	Data30	Data31	11
Data read/write	Data32	Data33	Data34	Data35	12
Data read/write	Data36	Data37	Data38	Data39	13
Data read/write	Data40	Data41	Data42	Data43	14
Data read/write	Data44	Data45	Data46	Data47	15

64 bytes

5.2.2.3.3. Read Binary Blocks

The "Read Binary Blocks" command is used for retrieving multiple "data blocks" from a MIFARE card. The data block/trailer block must be authenticated first before executing the "Read Binary Blocks" command.

Read Binary APDU Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Read Binary Blocks	0xFF	0xB0	0x00	Block Number	Number of Bytes to Read

Block Number (1 Byte):

The starting block

Number of Bytes to Read (1 Byte):

Multiple of 16 bytes for MIFARE 1K/4K or multiple of 4 bytes for MIFARE Ultralight

- Maximum 16 bytes for MIFARE Ultralight.
- Maximum 48 bytes for MIFARE 1K. (Multiple Blocks Mode; 3 consecutive blocks)
- Maximum 240 bytes for MIFARE 4K. (Multiple Blocks Mode; 15 consecutive blocks)

E.g.1: 0x10 (16 bytes) -> Read the starting block only. (Single Block Mode)

E.g.2: 0x40 (64 bytes) -> Read from the starting block to starting block + 3. (Multiple Blocks Mode)

***Note**: For safety reasons, the Multiple Block Mode is used for accessing Data Blocks only. The Trailer Block is not supposed to be accessed in Multiple Blocks Mode. Please use Single Block Mode to access the Trailer Block.*

Read Binary Block Response Format (Multiple of 4/16 + 2 bytes)

Response	Data Out		
Result	Data (Multiply of 4/16 Bytes)	SW1	SW2

Read Binary Block Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

Examples:

// Read 16 bytes from the binary block 0x04 (MIFARE 1K or 4K)

APDU = {0xFF 0xB0 0x00 0x04 0x10}

// Read 240 bytes starting from the binary block 0x80 (MIFARE 4K)

// Block 0x80 to Block 0x8E (15 blocks)

APDU = {0xFF 0xB0 0x00 0x80 0xF0}

5.2.2.3.4. Update Binary Blocks

The "Update Binary Blocks" command is used for writing multiple "data blocks" into a MIFARE card. The data block/trailer block must be authenticated first before executing the "Update Binary Blocks" command.

Update Binary APDU Format (Multiple of 16 + 5 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Update Binary Blocks	0xFF	0xD6	0x00	Block Number	Number of Bytes to Update	Block Data (Multiple of 16 Bytes)

Block Number (1 Byte):

The starting block to be updated.

Number of Bytes to Update (1 Byte):

- Multiple of 16 bytes for MIFARE 1K/4K or 4 bytes for MIFARE Ultralight.
- Maximum 48 bytes for MIFARE 1K. (Multiple Blocks Mode; 3 consecutive blocks)
- Maximum 240 bytes for MIFARE 4K. (Multiple Blocks Mode; 15 consecutive blocks)

E.g.1: 0x10 (16 bytes) -> Write the starting block only. (Single Block Mode)

E.g.2: 0x30 (48 bytes) -> Write from the starting block to starting block + 2. (Multiple Blocks Mode)

***Note**: For safety reasons, the Multiple Block Mode is used for accessing Data Blocks only. The Trailer Block is not supposed to be accessed in Multiple Blocks Mode. Please use Single Block Mode to access the Trailer Block.*

Block Data (Multiple of 16 + 2 bytes, or 6 bytes):

The data to be written into the binary block/blocks.

Update Binary Block Response Codes (2 bytes)

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

Examples:

// Update the binary block 0x04 of MIFARE 1K/4K with Data {0x00 0x01 .. 0x0F}

APDU = {0xFF 0xD6 0x00 0x04 0x10 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F}

// Update the binary block 0x04 of MIFARE Ultralight with Data {0x00 0x01 0x02 0x03}

APDU = {0xFF 0xD6 0x00 0x04 0x04 0x00 0x01 0x02 0x03}

5.2.2.3.5. Value Block Operation (INC, DEC, STORE)

The "Value Block Operation" command is used for manipulating value-based transactions e.g. increment a value of the value block etc.

Value Block Operation APDU Format (10 Bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Value Block Operation	0xFF	0xD7	0x00	Block Number	0x05	VB_OP	VB_Value (4 Bytes) {MSB ... LSB}

Block Number (1 Byte):

The value block to be manipulated.

VB_OP (1 Byte):

0x00 = Store the VB_Value into the block. The block will then be converted to a value block.

0x01 = Increment the value of the value block by the VB_Value. This command is only valid for value block.

0x02 = Decrement the value of the value block by the VB_Value. This command is only valid for value block.

VB_Value (4 Bytes):

The value used for value manipulation. The value is a signed long integer (4 bytes).

E.g. 1: Decimal "–4" = {0xFF, 0xFF, 0xFF, 0xFC}

VB_Value			
MSB			LSB
0xFF	0xFF	0xFF	0xFC

E.g. 2: Decimal "1" = {0x00, 0x00, 0x00, 0x01}

VB_Value			
MSB			LSB
0x00	0x00	0x00	0x01

Value Block Operation Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Value Block Operation Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

5.2.2.3.6. Read Value Block

The "Read Value Block" command is used for retrieving the value from the value block. This command is only valid for value block.

Read Value Block APDU Format (5 Bytes)

Command	Class	INS	P1	P2	Le
Read Value Block	0xFF	0xB1	0x00	Block Number	0x04

Block Number (1 Byte):

The value block to be accessed.

Read Value Block Response Format (4 + 2 bytes)

Response	Data Out		
Result	Value {MSB .. LSB}	SW1	SW2

Value (4 Bytes):

The value returned from the card. The value is a signed long integer (4 bytes).

E.g. 1: Decimal "–4" = {0xFF, 0xFF, 0xFF, 0xFC}

Value			
MSB			LSB
0xFF	0xFF	0xFF	0xFC

E.g. 2: Decimal “1” = {0x00, 0x00, 0x00, 0x01}

Value			
MSB			LSB
0x00	0x00	0x00	0x01

Read Value Block Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

5.2.2.3.7. Copy Value Block

The “Copy Value Block” command is used to copy a value from a value block to another value block.

Copy Value Block APDU Format (7 Bytes)

Command	Class	INS	P1	P2	Lc	Data In	
Copy Value Block	0xFF	0xD7	0x00	Source Block Number	0x02	0x03	Target Block Number

Source Block Number (1 Byte):

The value of the source value block will be copied to the target value block.

Target Block Number (1 Byte):

The value block to be restored. The source and target value blocks must be in the same sector.

Copy Value Block Response Format (2 bytes)

Response	Data Out	
Result	SW1	SW2

Copy Value Block Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

5.2.2.4. Access PCSC Compliant Tags (ISO14443-4)

All ISO14443-4 compliant cards (PICCs) understand the ISO 7816-4 APDUs. The AMR220-C1 just has to communicate with the ISO 14443-4 compliant cards by exchanging ISO 7816-4 APDUs and responses. The AMR220-C1 will handle the ISO 14443 Parts 1-4 Protocols internally.

MIFARE 1K, 4K, MINI and Ultralight tags are supported through the T=CL emulation. Simply treat the MIFARE tags as standard ISO14443-4 tags.

For more information, please refer to **PICC Commands (T=CL Emulation) for MIFARE 1K/4K MEMORY Cards**.

ISO 7816-4 APDU Format

Command	Class	INS	P1	P2	Lc	Data In	Le
ISO 7816-4 Command					Length of the Data In		Expected length of the Response Data

ISO 7816-4 Response Format (Data + 2 bytes)

Response	Data Out		
Result	Response Data	SW1	SW2

Common ISO 7816-4 Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x63	0x00	The operation failed.

Typical sequence may be:

- Present the Tag and Connect
- Read / Update the memory of the tag

Operation Example:

Step 1. Connect the Tag

The ATR of the tag is 0x3B 0x88 0x80 0x01 0x00 0x00 0x00 0x00 0x33 0x81 0x81 0x00 0x3A

In which it is an ISO14443-4 Type B tag with,

The Application Data of ATQB = 0x00 0x00 0x00 0x00

Protocol information of ATQB = 0x33 0x81 0x81

Step 2. Send an APDU, e.g. Get Challenge

CMD: 0x00 0x84 0x00 0x00 0x08

RSP: 0x1A 0xF7 0xF3 0x1B 0xCD 0x2B 0xA9 0x58 [0x90 0x00]

***Note**: For ISO14443-4 Type A tags, the ATS can be obtained by using the APDU "0xFF 0xCA 0x01 0x00 0x00"*

5.2.2.5. Access FeliCa Tags

For FeliCa access, the command differs from the specifications of PCSC Compliant Tags and MIFARE cards. It follows FeliCa specifications, with a header added. The following is the format:

FeliCa Command Format

Command	Class	INS	P1	P2	Lc	Data In
FeliCa Command	0xFF	0x00	0x00	0x00	Length of the Data In	FeliCa Command (start with Length Byte)

FeliCa Command:

Please refer to FeliCa Card specifications

Examples:

e.g.1 Polling Command

= {0x06, 0x00, 0xFF, 0xFF, 0x00, 0x00}

In which
0x00 = Polling Command Code
0xFF 0xFF = System Code

e.g.2 Read Without Encryption Command

= {0x10 0x06 0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03 0x01 0x09 0x01 0x01 0x80 0x00}

In which
0x06 = Read Without Encryption Command Code
0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03 = Felica IDm (depend on card)
0x09 0x01= Service Code
0x80 0x00 = Block

FeliCa Response Format (Data + 2 bytes)

Response	Data Out		
Result	Response Data	SW1	SW2

Response Codes

Results	SW1	SW2	Meaning
Success	0x90	0x00	The operation is completed successfully.
Error	0x67	0x00	Length error
	0x64	0x01	The operation failed.

Operation Example:

Step 1. Connect the FeliCa

The ATR = 0x3B 0x8F 0x80 0x01 0x80 0x4F 0x0C 0xA0 0x00 0x00 0x03 0x06 0x11 0x00 0x3B 0x00 0x00 0x00 0x00 0x42

In which 11 00 3B = FeliCa

Step 2. Read FeliCa IDM

CMD = 0xFF 0xCA 0x00 0x00 0x00

RSP = [IDM (8bytes)] 0x90 0x00

e.g. the FeliCa IDM = **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03**

Step 3. FeliCa Command Access (Example to use Read without Encryption Command)

CMD = 0xFF 0x00 0x00 0x00 0x10 0x10 0x06 **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03** 0x01 0x09 0x01 0x01 0x80 0x00

In which

Felica Command = 0x10 0x06 **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03** 0x01 0x09 0x01 0x01 0x80 0x00

Felica IDm = 0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03

RSP = 0x1D 0x07 **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03** 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x90 0x00

In which

Response Code = 0x07

Felica IDm = 0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03

Status Flag = 0x00 0x00

Block Data = 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

5.3. Escape Command

The Escape command is used to control the peripherals or special operation.

The commands are sent through SPH_to_RDR_ExEscape or PCSC SCardControl with dwControlCode = SCARD_CTL_CODE(3500).

5.3.1. Get Firmware Version

The “Get Firmware Version” command is used to get the AMR220-C1’s firmware message.

Get Firmware Version Format (4Bytes)

Command	Class	INS	P1	P2
Get Firmware Version	0xFC	0x00	0xA1	0xFF

Get Firmware Version Response Format (3 Bytes + Firmware Message Length)

Response				Data Out
Result	0x00	0x30	0x30	Firmware Version

e.g. Response = 0x00 0x30 0x30 0x31 0x2E 0x30 0x2E 0x31 0x34

Firmware Version (HEX) =0x31 0x2E 0x30 0x2E 0x31 0x34

Firmware Version (ASCII) = “1.0.14”

5.3.2. Sleep Mode Option

The “Set Sleep Time Interval” command is used to get/set the time interval before entering sleep mode for the AMR220-C1.

Set Sleep Time Interval Command Format (6 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Sleep Time Interval	0xE0	0x00	0x00	0x48	0x01	Time

Or

Get Sleep Time Interval Command Format (5 Bytes)

Command	Class	INS	P1	P2	Lc
Sleep Time Interval	0xE0	0x00	0x00	0x48	0x00

Set Sleep Time Interval Response Format (6 Bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	0xE1	0x00	0x00	0x00	0x01	Time

Where:

Time 1 byte (units in second)

Data In = 01 to FF

Default = 0x78 (120 sec)

5.3.3. Antenna Field Control

The “Antenna Field Control” command is used to control the Antenna Field.

***Note**: The antenna field will be affected by Auto Polling.*

Antenna Field Control Command Format (6 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Antenna Field Control	0xE0	0x00	0x00	0x41	0x01	0x00 – Antenna Off 0x01 – Antenna On

Antenna Control Response Format (6 Bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	0xE1	0x00	0x00	0x00	0x01	0x00 – Antenna Off 0x01 – Antenna On

5.3.4. Automatic PICC Polling

The “Automatic PICC Polling” command is used to set the reader’s polling mode when USB is used for communication.

Whenever the AMR220-C1 is connected to the PC, the PICC polling function will start the PICC scanning to determine if a PICC is placed on / removed from the built-antenna.

We can send a command to disable the PICC polling function. To meet the energy saving requirement, special modes are provided for turning off the antenna field whenever the PICC is inactive, or no PICC is found. The reader will consume less current in power saving mode.

***Notes**:*

1. *The Auto Polling feature is for USB mode ONLY.*
2. *The setting will be set to default once AMR220-C1 is reset.*

Set Automatic PICC Polling Command Format (6 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Automatic PICC Polling	0xE0	0x00	0x00	0x23	0x01	Polling Setting

or

Get Automatic PICC Polling Command Format (5 Bytes)

Command	Class	INS	P1	P2	Lc
Automatic PICC Polling	0xE0	0x00	0x00	0x23	0x00

Automatic PICC Polling Response Format (6 Bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	0xE1	0x00	0x00	0x00	0x01	Polling Setting

Polling Setting (1 Byte):

Polling Setting	Description	Description
Bit 0	Auto PICC Polling	1 = Enable; 0 =Disable
Bit 1	Turn off Antenna Field if no PICC found	1 = Enable; 0 =Disable
Bit 2	Turn off Antenna Field if the PICC is inactive.	1 = Enable; 0 =Disable
Bit 3	RFU	-
Bit 5 ... 4	PICC Poll Interval for PICC	<Bit 5 – Bit 4> <0 – 0> = 250 ms <0 – 1> = 500 ms <1 – 0> = 1000 ms <1 – 1> = 2500 ms
Bit 6	RFU	-
Bit 7	Enforce ISO14443A Part 4	1= Enable; 0= Disable.

**Default value of Polling Setting = 0x8Bh*

Notes:

1. *It is recommended to enable the option "Turn Off Antenna Field if the PICC is inactive", so that the Inactive PICC will not be exposed to the field all the time, preventing it from warming up.*
2. *The longer the PICC Poll Interval, the more energy may be efficiently saved. However, the response time of PICC Polling will become longer.*
3. *The reader will activate the ISO14443A-4 mode of the "ISO14443A-4 compliant PICC" automatically. Type B PICC will not be affected by this option.*
4. *The JCOP30 card comes with two modes: ISO14443A-3 (MIFARE 1K) and ISO14443A-4 modes. The application has to decide which mode should be selected once the PICC is activated.*

5.3.5. PICC Operating Parameter

The “PICC Operating Parameter” command is used to set the Automatic Polling’s Detect Card Type.

Notes:

1. *The Auto Polling feature is for USB mode ONLY.*
2. *The setting will be set to default once the AMR220-C1 is reset.*

Set PICC Operating Parameter Command Format (6 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
PICC Operating Parameter	0xE0	0x00	0x00	0x20	0x01	Operating Parameter

Or

Get PICC Operating Parameter Command Format (5 Bytes)

Command	Class	INS	P1	P2	Lc
PICC Operating Parameter	0xE0	0x00	0x00	0x20	0x00

PICC Polling Operating Parameter Response Format (6 Bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	0xE1	0x00	0x00	0x00	0x01	Operating Parameter

Operating Parameter (1 Byte):

Card Type	Parameter	Description	Option
Bit0	ISO14443 Type A	The tag types to be detected during PICC Polling.	1 = Detect 0 = Skip
Bit1	ISO14443 Type B		1 = Detect 0 = Skip
Bit2	Felica 212kbps		1 = Detect 0 = Skip
Bit3	Felica 424kbps		1 = Detect 0 = Skip
Bit4 - 7	RFU	RFU	RFU

**Default value of Card Type = 0x0F*

5.3.6. Buzzer Control

The “Buzzer Control” command is used to control the buzzer sound output.

Buzzer Control Format #1 (6 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
Buzzer Control	0xE0	0x00	0x00	0x28	0x01	Buzzer on Duration

Buzzer on Duration (1 Byte):

0x00 = Turn OFF

0x01 to 0xFF = Duration (unit: 10ms) with Freq = 1500Hz

Or

Buzzer Control Format #2 (8 Bytes)

Command	Class	INS	P1	P2	Lc	Data In		
Buzzer Control	0xE0	0x00	0x00	0x28	0x03	Buzzer on Duration	Buzzer off Duration	Repeat Count

Buzzer on Duration (1 Byte):

0x00 = Turn OFF

0x01 to 0xFF = Turn ON Duration (unit: 10ms) with Freq = 1500Hz

Buzzer off Duration (1 Byte):

0x00 = Turn ON with Freq = 1500Hz

0x01 to 0xFF = Turn OFF Duration (unit: 10ms)

Repeat Count (1 Byte):

Number of Turn ON and Turn OFF pattern repeat

Or

Buzzer Control Format #3 (12 bytes)

Command	Class	INS	P1	P2	Lc	Data In			
Buzzer Control	0xE0	0x00	0x00	0x28	0x07	Buzzer on Duration	Buzzer off Duration	Repeat Count	Frequency (MSB … LSB)

Buzzer on Duration (1 Byte):

0x00 = Turn OFF

0x01 to 0xFF = Turn ON Duration (unit: 10ms)

Buzzer off Duration (1 Byte):

0x00 = Turn ON

0x01 to 0xFF = Turn OFF Duration (unit: 10ms)

Repeat Count (1 Byte):

Number of Turn ON and Turn OFF pattern repeat

Frequency (4Bytes):

Frequency for the buzzer output
Frequency = 1500 -> 1500Hz
Frequency = 750 -> 750Hz
Frequency = other value -> RFU

Buzzer Control Response Format (6 Bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	0xE1	0x00	0x00	0x00	0x01	0x00

5.3.7. LED Control

The “LED Control” command is used to control the LEDs Output.

Set LED Control Command Format (6 Bytes)

Command	Class	INS	P1	P2	Lc	Data In
LED Control	0xE0	0x00	0x00	0x29	0x01	LED Status

Or

Get LED Control Command Format (5Bytes)

Command	Class	INS	P1	P2	Lc
LED Control	0xE0	0x00	0x00	0x29	0x00

LED Control Response Format (6 Bytes)

Response	Class	INS	P1	P2	Le	Data Out
Result	0xE1	0x00	0x00	0x00	0x01	LED Status

LED Status (1 Byte):

Polling Setting	Description	Description
Bit 0	Green 1 LED	1 = ON; 0 = OFF
Bit 1	Green 2 LED	1 = ON; 0 = OFF
Bit 2	Green 3 LED	1 = ON; 0 = OFF
Bit 3	Green 4 LED	1 = ON; 0 = OFF
Bit 4 ... 7	RFU	RFU