AsReader Dock SDK Manual

AsReader Dock SDK Manual V1.2

For ASX-300R,ASX-301R, ASX-510R, ASX-520R, ASR-010D, ASR-020D, ASR-030D, ASR-031D

Asterisk, Inc.
2018-07-20

Modification

No.	Version	Modified Content	Date
1	1.2	Initial version	2018/07/20

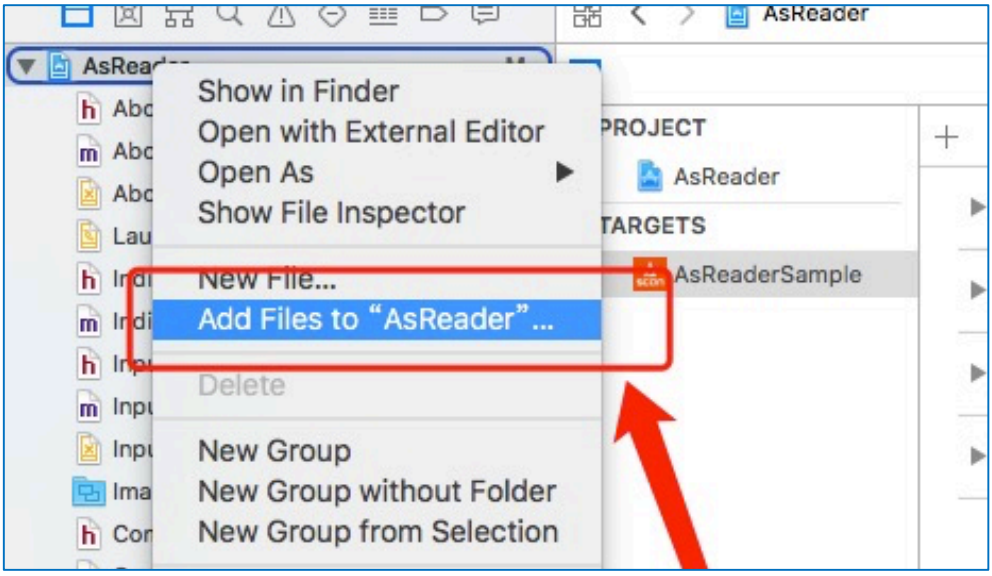
Contents

1. SDK Usage

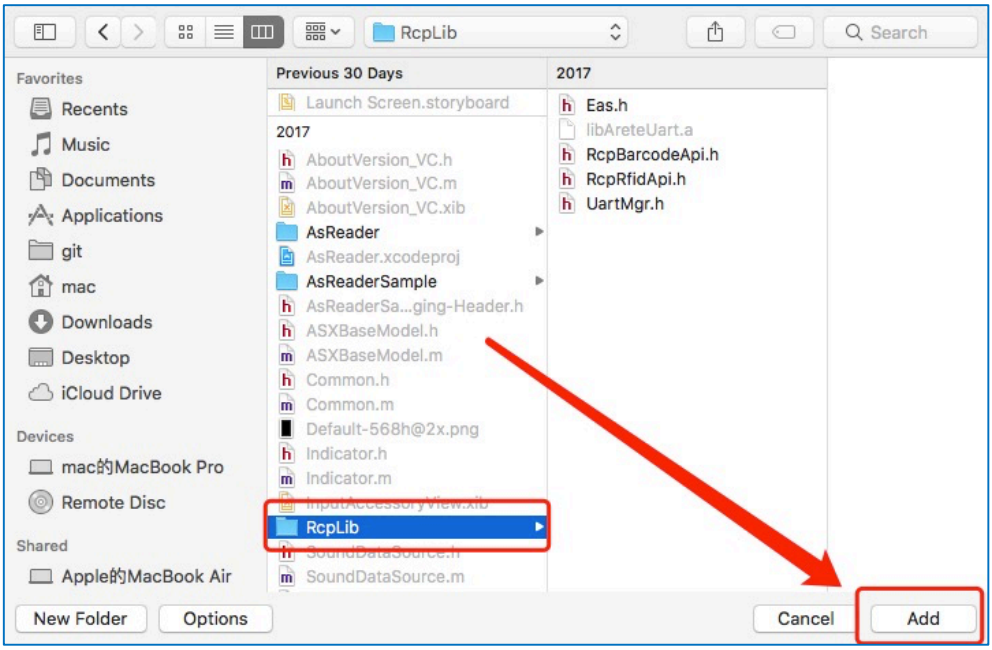
1.1. Import header files of the SDK to project

· #import “Eas.h”
· #import “UarMgr.h”
· #import “RcpBarcodeApi.h”
· #import “RcpRfidApi.h”

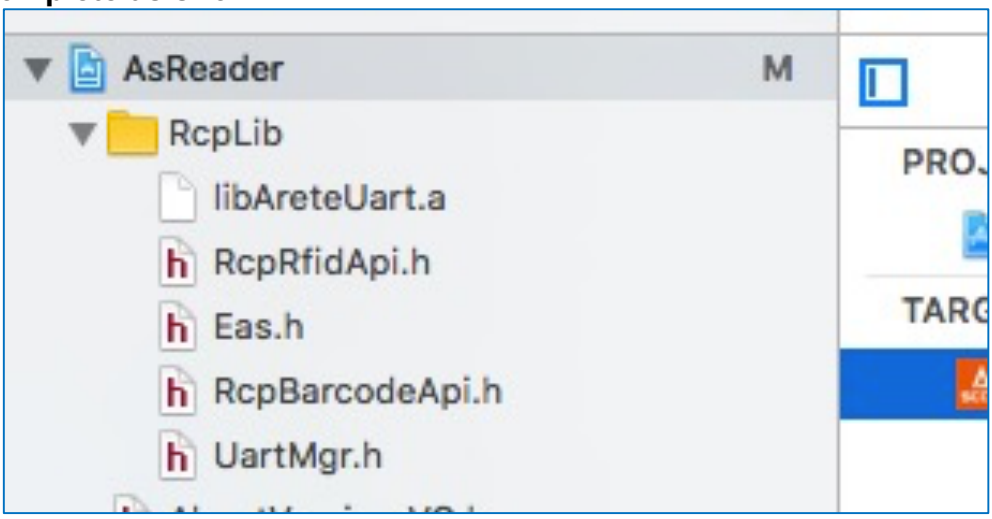
1.1.1 “Add Files to”



1.1.2 Select “Add”

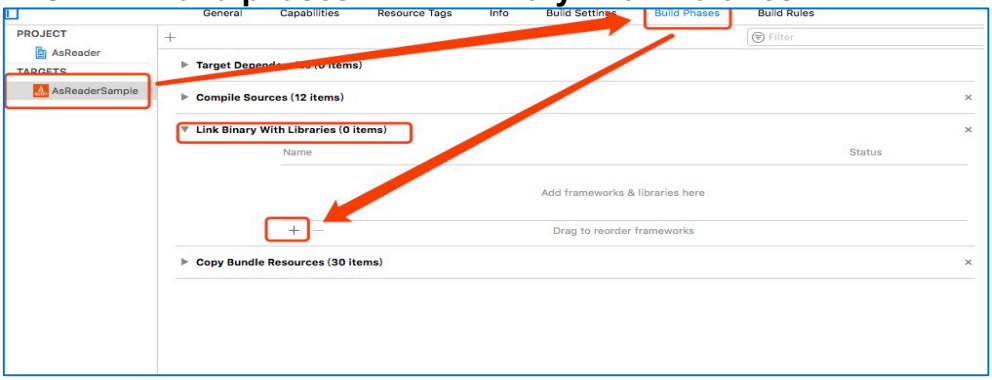


1.1.3 Complete as shown

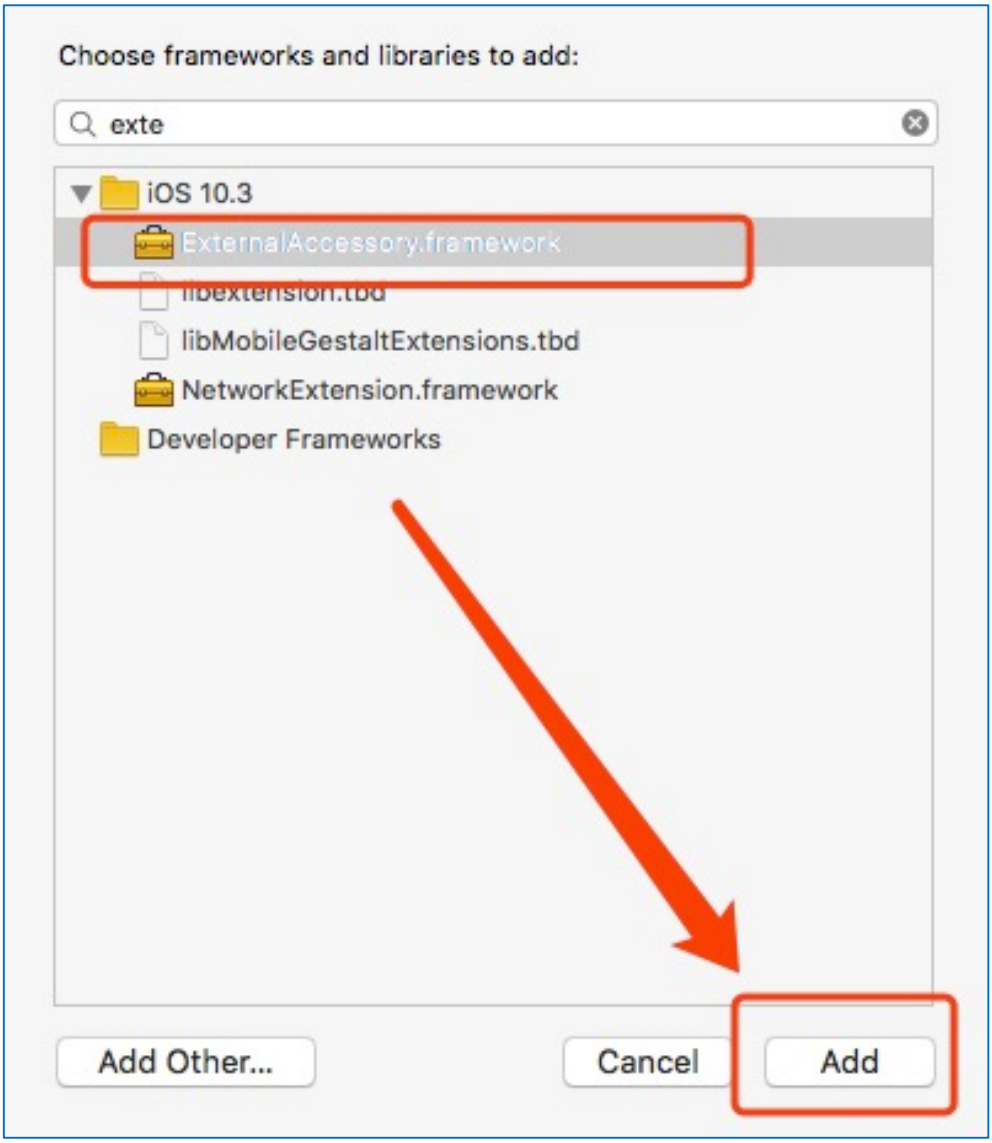


1.2. Add ExternalAccessory.framework

1.2.1 TARGET -> Build phases -> Link Binary With Libraries



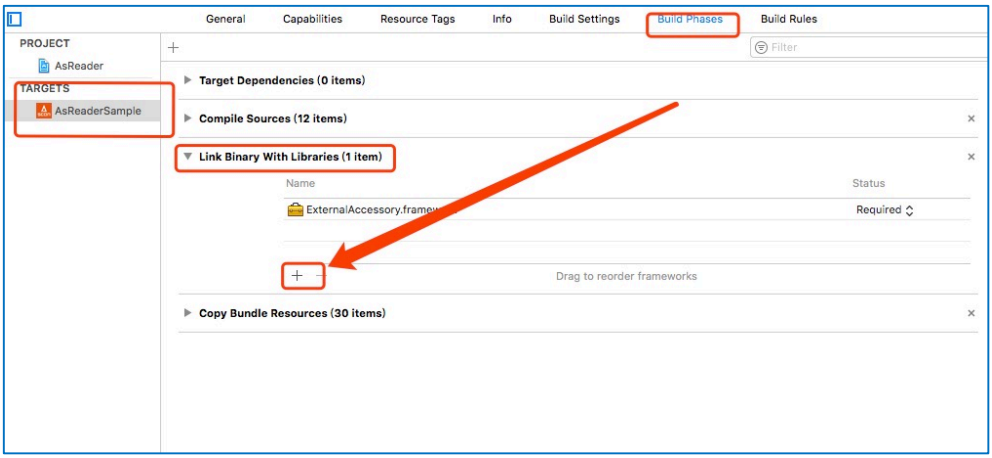
1.2.2 Select “Add”



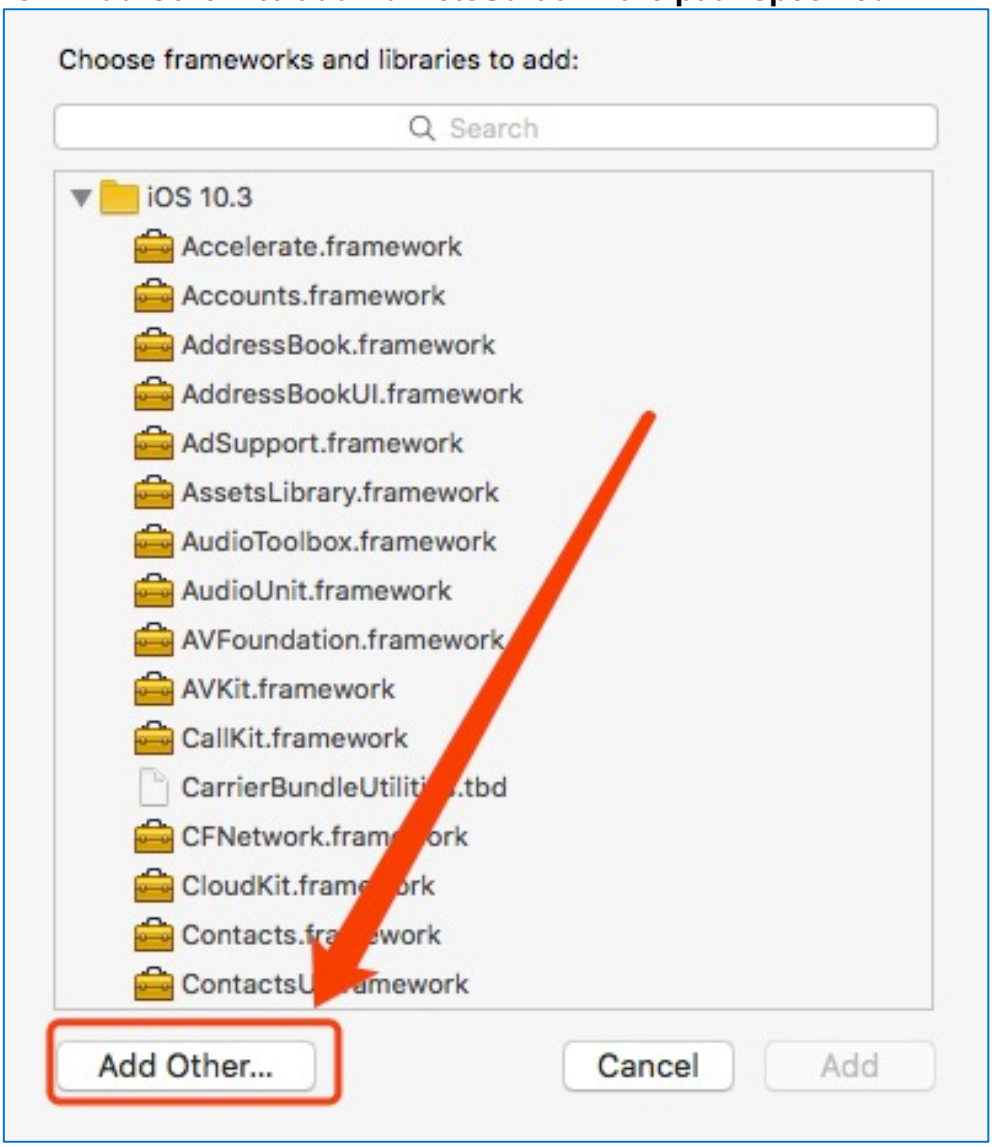
1.2.3 Verify that the “ExternalAccessory.framework” has been added.

1.3. Import libAreteUart.a

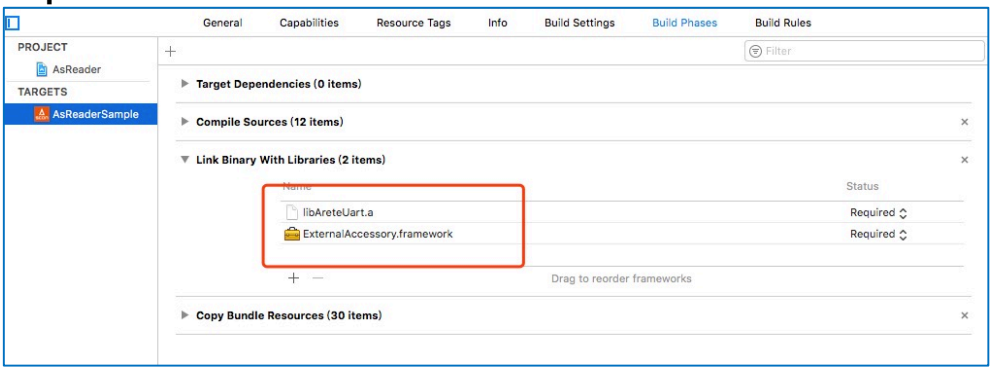
1.3.1 Add libAreteUart.a



1.3.2 Click “Add Other” to add libAreteUart.a in the path specified



1.3.3 Complete as shown:

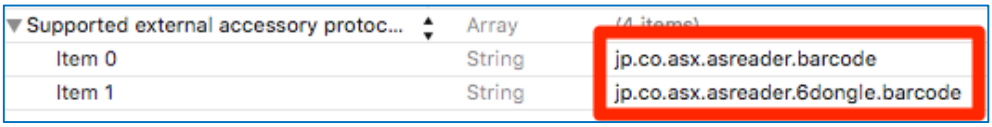


1.4. Add AsReader protocol

In **Supported external accessory protocols** of plist, add the corresponding protocol to the following devices.

- ASX-510R,520R:jp.co.asx.asreader.barcode

- ASR-010D,020D:jp.co.asx.asreader.6dongle.barcode
- ASX-300R,ASX-301R:jp.co.asx.asreader.rfid
- ASR-030R,ASR-031R:jp.co.asx.asreader.6dongle.rfid



1.5. Use SDK In Class

Import the SDK Class header file into the Objective C project, the following is one example:

#import “RcpBarcodeApi.h”

1.6. Precaution

If you need to support C++ while using the SDK in Objective C, change the imported SDK header file suffix from *.m to *.mm, or import the libc++ library and compile.

2. BarcodeApi Class

Supported AsReader: ASX-510R,ASX-520R,ASR-010D,ASR-020D.

2.1. init

- (id)init;

Description: Initialize RcpBarcodeApi Class.
Return value: initialization success:instance of RcpBarcodeApi Class
initialization:nil

2.2. open

- (BOOL)open;

Description: Open connection to the reader.
Return value:
YES: connected
NO: disconnected

2.3. isOpened

- (BOOL)isOpened;

Description: Receive connection status of the reader.
Return value:
YES: connected
NO: disconnected

2.4. close

- (void)close;

Description: Close connection to the reader.

2.5. startReadBarcodes

- (BOOL)startReadBarcodes:(uint8_t)mtnu mtime:(uint8_t)mtime repeatCycle:(uint16_t)repeatCycle;

Description: The reader starts scanning barcodes.
Parameter:
mtnu: 0x00, Maximum number of tags to read
mtime: 0x00, Maximum elapsed time to read tags (sec)
repeatCycle: How many times the reader performs an inventory round
Return value:
YES: connected
NO: disconnected

2.6. setReaderPower

- (BOOL)setReaderPower:(BOOL)on;

Description: Set reader power on/off.

Parameter: on
open: YES
close: NO
Return value:
YES: connected
NO: disconnected

2.7. setBeep

```
- (void)setBeep:(uint8_t)beepOn
        setVibration:(uint8_t)vibrationOn
        setIllumination:(uint8_t)illuminationOn;
```

Description: Send the command to the reader to set beep, vibration, and illumination settings.
Parameter:
beepOn: beep, On:0x01/Off:0x00
vibrationOn: vibration, On:0x01/Off:0x00
illuminationOn: illumination, On:0x01/Off:0x00

2.8. getSDKVersion

```
- (NSString*)getSDKVersion;
```

Description: Get SDK version.
Return value: version number (for example:1.3.3)

3. BarcodeDelegate Class

3.1. readerConnected

- (void)readerConnected:(uint8_t)status;

Description: This method is called after calling “setReaderPower” and “setReaderPowerOnWithBeep”.
Parameter: status
poweron success: 0xFF
poweron failure: 0x00
Notes: Do not call other methods before this method is called.

3.2. pluggedBarcode

- (void)pluggedBarcode:(BOOL)plug;

Description: This method is called when the reader’s connection status changes.
Parameter:plug
YES: success
NO: failure

3.3. barcodeStringReceived

- (void)barcodeStringReceived:(NSString *)barcode;

Description: Call the method “startReadBarcodes”, press the reader’s trigger button, and this method will be called when barcode data is received.
Parameter: barcode: barcode data(NSString)

3.4. batteryChargeReceived

- (void)batteryChargeReceived:(int)battery;

Description: Receive current battery level when the reader is connected (receive battery level every 10s).
Parameter:
battery:
0,25,50,75,100 (value is shown as a percentage, for example:75 current battery is 75%). the effective battery level for each percentage:
25%: 1～25%
50%: 26～50%
75%: 51～75%
100%: 76～100%

4. RfidApi Class

Supported AsReader: ASX-300R,ASX-301R,ASR-030D,ASR-031D.

4.1. getSDKVersion

- (NSString*)getSDKVersion;

Description: Get SDK version.
Return value: version number (for example:1.3.3)

4.2. init

- (id)init;

Description: Initiate RcpRfidApi.
Return value:success:instance of RcpBarcodeApi
failure:nil

4.3. open

- (BOOL)open;

Description: Open connection to the reader.
Return value:
YES: connected
NO: disconnected

4.4. isOpened

- (BOOL)isOpened;

Description: Receive connection status of the reader.
Return value:
YES: success
NO: failure

4.5. close

- (void)close;

Description: Close connection to the reader.

4.6. setReaderPower

- (BOOL)setReaderPower:(BOOL)on connectedBeep:(BOOL)connectedBeep;

Description: Set reader power on/off.
Parameter:
on:YES:on/NO:off
connectedBeep:YES:on/NO:off
Return value:

YES: success
NO: failure

4.7. setBeep

```
- (BOOL)setBeep:(uint8_t)beepOn
     setVibration:(uint8_t)vibrationOn
     setIllumination:(uint8_t)illuminationOn;
```

Description: Set reader beep, vibration, and illumination settings.
Parameter:
beepOn:On:0x01/Off: 0x00
vibrationOn: On:0x01/Off: 0x00
illuminationOn: On:0x01/Off: 0x00

4.8. startReadTags

```
- (BOOL)startReadTags:(uint8_t)mtnu mtime:(uint8_t)mtime
  repeatCycle:(uint16_t)repeatCycle;
```

Description: The reader starts reading tags.
Parameter:
mtnu: 0x00, Maximum number of tags to read
mtime: 0x00, Maximum elapsed time to read tags (sec)
repeatCycle: How many times the reader performs an inventory round
Return value:
YES: success
NO: failure

4.9. startReadTagsWithRssi

```
– (BOOL)startReadTagsWithRssi:(uint8_t)mtnu
                      mtime:(uint8_t)mtime
                repeatCycle:(uint16_t)repeatCycle;
```

Description: Start reading tags with RSSI. Parameters show the stop conditions for reading.
Parameter:
mtnu: 0x00, Maximum number of tags to read
mtime: 0x00, Maximum elapsed time to read tags (sec)
repeatCycle: How many times the reader performs an inventory round
Return value:
YES: connected
NO: disconnected

4.10. stopReadTags

```
- (BOOL)stopReadTags;
```

Description: Stop reading RFID tags.
Return value:
YES: success
NO: failure

4.11. getChannel

```
- (BOOL)getChannel;
```

Description: Send the "Get current RF Channel" command to the reader to get the RF channel. This command is valid only for non-FH mode.

Return value:

YES: success

NO: failure

4.12. setChannel

```
- (BOOL)setChannel:(uint8_t)channel
    channelOffset:(uint8_t)channelOffset;
```

Description: Send the "Set current RF Channel" command to the reader to set the RF channel. This command is valid only for non-FHSS mode.

Parameter:

channel:Channel number. The range of channel number depends on regional settings.

channelOffset:Channel number offset for miller subcarrier.

Return value:

YES: success

NO: failure

4.13. getFhLbtParam

```
- (BOOL)getFhLbtParam;
```

Description: Send the "Get FH and LBT Parameters" command to the reader to get FH and LBT control.

Return value:

YES: success

NO: failure

4.14. setFhLbtParam

```
- (BOOL)setFhLbtParam:(uint16_t)readTime
        idleTime:(uint16_t)idleTime
        carrierSenseTime:(uint16_t) carrierSenseTime
        rfLevel:(uint16_t)rfLevel
        frequencyHopping:(uint8_t)frequencyHopping
        listenBeforeTalk:(uint8_t)listenBeforeTalk
        continuousWave:(uint8_t)continuousWave;
```

Description: Send the "Set FH and LBT Parameters" command to the reader to set FH and LBT Parameters.

Parameter:

readTime: read time(ms)

idleTime: idle time(ms)

carrierSenseTime: carrier sense(ms)

rfLevel: Target RF power(-dBm X 10)

frequencyHopping: enable:0x01or over/disable:0x00
listenBeforeTalk: enable:0x01 or over/disable:0x00
continuousWave: enable:0x01/disable:0x00

Return value:

YES:success
NO:failure

```
- (BOOL)getOutputPowerLevel;
```

4.15. getOutputPowerLevel

Description: To get the current, minimum, and maximum Tx power level. (Assign the power value obtained to the object of the CommonReaderInfo class by the delegate method txPowerLevelReceived.)

Return value:

YES: success
NO: failure

4.16. setOutputPowerLevel

```
- (BOOL)setOutputPowerLevel:(uint16_t)power;
```

Description: Set the current Tx power level

Parameter:

power:Tx power. （The Tx power range of Japanese version : 18 ~ 24dBm, that of non-japanese version: 18 ~ 25dBm).

Return value:

YES: success
NO: failure

4.17. readFromTagMemory

```
- (BOOL)readFromTagMemory:(uint32_t)accessPassword
        epc:(NSData*)epc
        memoryBank:(uint8_t)memoryBank
        startAddress:(uint16_t)startAddress
        dataLength:(uint16_t)dataLength;
```

Description: To read the Type C tag data of the specified memory.

Parameter:

accessPassword: The access password
epc: The target tag
memoryBank: RFU (0x00), EPC (0x01), TID (0x02), User (0x03)
startAddress: The starting address
dataLength: the Length of the data

Return value:

YES: success
NO: failure

4.18. getSession

```
- (BOOL)getSession;
```

Description: Send the "Get Session" command to the reader to get the current session.
Return value:
YES: success
NO: failure

4.19. setSession

```
- (BOOL)setSession:(uint8_t)session;
```

Description: Send the "Set Session" command to the reader to set the current session.
Parameter:session:S0:0x00/S1:0x01/S2:0x02/S3:0x03/Dev.mode:0xF0
Return value:
YES: success
NO: failure

4.20. getAnticollision

```
- (BOOL)getAnticollision;
```

Description: Send the "Get Anti-Collision Mode" command to the reader to get the Anti-collision algorithm.
Return value:
YES: success
NO: failure

4.21. setAnticollision

```
– (BOOL)setAnticollision:(uint8_t)mode
  qStart:(uint8_t)qStart
  qMax:(uint8_t)qMax
  qMin:(uint8_t)qMin;
```

Description: Send the "Set Anti-Collision Mode" command to the reader to set the Anti-collision algorithm.
Parameter:
mode:Anti-collision Mode (8-bit), fixed Q: 0x00/Dynamic Q: 0x01
qStart: starting Q
qMax: maximum Q
qMin: minimum Q
Return value:
YES: connected
NO: disconnected

4.22. writeToTagMemory

```
- (BOOL)writeToTagMemory:(uint32_t)accessPassword
        epc:(NSData*)epc
        memoryBank:(uint8_t)memoryBank
        startAddress:(uint16_t)startAddress
        dataToWrite:(NSData*)dataToWrite;
```

Description: Send the "Write Type C Tag Data" command to the reader to write type C tag data.

Parameter:

accessPassword: access password
epc: target tag's EPC.
memory bank:memory bank
startAddress:starting address
dataToWrite: data to write

Return value:

YES: connected
NO: disconnected

4.23. killTag

```
- (BOOL)killTag:(uint32_t)killPassword
            epc:(NSData*)epc;
```

Description: Send the "Kill Type C Tag" command to the reader to kill a tag.
Note:Must set kill password before killing tag.

Parameter:

killPassword: kill password. If KP filed set to 0x00000000, the 'Kill Type C Tag' command will not work. The target tag will ignore it.
epc:target tag's epc.

Return value:

YES: success
NO: failure

4.24. lockTagMemory

```
- (BOOL)lockTagMemory:(uint32_t)accessPassword
                  epc:(NSData*)epc
            lockData:(uint32_t)lockData;
```

Description: Send the "Lock Type C Tag" command to the reader to lock an indicated memory bank in the tag.

Notes:Must set access password before locking tag.

Parameter:

accessPassword: access password if memory bank was password protected. Otherwise, set AP filled as 0x00000000.
epc:target tag's epc.
lockData:Lock mask and action flags. Pad 4-bit zeros (dummy) to the left of 20-bit lock mask and associated action flags.

Return value:

YES: success
NO: failure

4.25. setStopConditionMtnu

```
-(BOOL) setStopConditionMtnu:(uint8_t)mtnu
                    setMtime:(uint8_t)mtime
               setRepeatCycle:(uint16_t)repeatCycle;
```

Description: Send the "Set Stop Condition" command to the reader to set the stop point of start-auto-read.
Parameter:
mtnu : Maximum number of tags to read
mtime : Maximum elapsed time to read tags (sec)
repeatCycle : How many times the reader performs an inventory round
Return value:
YES:success
NO:failure

4.26. setOptimumFrequencyHoppingTable

```
- (BOOL)setOptimumFrequencyHoppingTable;
```

Description: Set optimum frequency hopping table.
Return value:
YES: success
NO: failure

4.27. GetFrequencyHoppingMode

```
-(BOOL) GetFrequencyHoppingMode;
```

Description: Get frequency hopping mode.
Return value:
YES: success
NO: failure

4.28. updateRegistry

```
- (BOOL)updateRegistry;
```

Description: Update registry.
Return value:
YES: success
NO: failure

4.29. writeToTagMemory

```
- (BOOL)writeToTagMemory:(NSData*)epc
      dataToWriteAscii:(NSString*)dataToWrite;
```

Description: Send the "Write Type C Tag Data" command to the reader to write type C tag data.
Parameter:
epc: target tag's EPC
dataToWrite:data to write
Return value:

YES: connected
NO: disconnected

4.30. startReadTagsRFM

- (BOOL)startReadTagsRFM:(uint8_t)codeType mtnu:(uint8_t)mtnu mtime:(uint8_t)mtime repeatCycle:(uint16_t)repeatCycle;

Description: Start to read the RFID temperature tag / humidity tag.

Parameter:

codeType: tag type (temperature tag: 0x03, humidity tag: 0x02)
mtnu: Maximum number of tags to read
mtime: Maximum time to read, in units: s
repeatCycle: Number of read iterations

Return value:

YES: connected
NO: disconnected

5. RfidDelegate Class

Supported Asreader: ASX-300R,ASX-301R,ASR-030D,ASR-031D.

5.1. pluggedRfid

```
- (void)pluggedRfid:(BOOL)plug;
```

Description: This method is called when the reader's connection status changes.
Parameter:
YES: connection success
NO: connection failure

5.2. pcEpcReceived

```
- (void)pcEpcReceived:(NSData *)pcEpc;
```

Description: This function is called when tag data is received.
Parameter:pcEpc:pcepc data

5.3. pcEpcRssiReceived

```
- (void)pcEpcRssiReceived:(NSData *)pcEpc
                    rssi:(int8_t)rssi;
```

Description: This function is called when a tag's pcEpc with rssi is received.
Parameter:
pcEpc:pcepc data
rssi: rssi data

5.4. readerConnected

```
- (void)readerConnected:(uint8_t)status;
```

Description: Notification about "Power Reset" from module. It is a function that is called when changes the reader's connection information.
Parameter:status(connected:0xFF/disconnected:0x00.)

5.5. readerConnected

```
- (void)readerConnected;
```

Description: Notification from the module about "Power Reset". This function is called when the reader's connection status changes.

5.6. errReceived

```
- (void)errReceived:(uint8_t)errCode;
```

Description: Response to an invalid command.
Parameter:errCode: payload (error code, command code, sub error code).

5.7. errDetailReceived

```
- (void)errDetailReceived:(NSData *)errCode;
```

Description: Receive detailed error information. This method is called when the executed command is incorrect.
Parameter: errCode: payload (error code, command code, sub error code)

5.8. frequencyHoppingModeReceived

```
- (void)frequencyHoppingModeReceived:(uint8_t)statusCode;
```

Description: Response to "Get FreqHopping Table". This function is called when a response code to "Get Reader Information" is received.
Parameter:statusCode(connected:0xFF/disconnected:0x00)

5.9. regionReceived

```
- (void)regionReceived:(uint8_t)region;
```

Description: This function is called when a response code to "Get Region" is received.
Parameter: region:korea(0x11), North america(0x21), US(0x22), Europe(0x31), Japan(0x41), China1(0x51), China2(0x52), Brazil(0x61)

5.10. channelReceived

```
- (void)channelReceived:(uint8_t)channel
channelOffset:(uint8_t)channelOffset;
```

Description: This function is called when a response code to "Get current RF Channel" is received.
Parameter:

channel: channel of rfid module
channeloffset: channel offset of rfid module

5.11. fhLbtReceived

```
- (void)fhLbtReceived:(NSData *)fhLb;
```

Description: This function is called when a response code to "Get FhLbt Param" is received.
Parameter: fhLb: FH and LBT

5.12. tagMemoryReceived

```
- (void)tagMemoryReceived:(NSData *)data;
```

Description: This function is called when a response code to "Write to TagMemory" is received.
Parameter:data:memory information of tag

5.13. anticolParamReceived

```
- (void)anticolParamReceived:(uint8_t)mode start:(uint8_t)start
max:(uint8_t)max min:(uint8_t)min;
```

Description: This function is called when a response code to "Get Anti-Collision Mode" is received.
Parameter:

mode:fixed Q: 0x00/dynamic Q:0x01
max: maximum Q
min:minimum Q

5.14. batteryChargeReceived

- (void)batteryChargeReceived:(int)battery;

Description: Get current battery level when the reader is successfully connected (receive battery every 10s).
Parameter:

battery:0,25,50,75,100(value is shown as a percentage, for example:75 current battery level is 75%), the effective battery level for each percentage:

25%: 1～25%
50%: 26～50%
75%: 51～75%
100%: 76～100%

5.15. startedReadTags

- (void)startedReadTags:(uint8_t)statusCode;

Description: This function is called when the reader sends a response code to "startReadTags".
Parameter:statusCode:(success: 0x00/ failure: non-0x00)

5.16. didSetOutputPowerLevel

- (void)didSetOutputPowerLevel:(uint8_t)status;

Description: This function is called when a response code to "Set Tx Power Level" is received.
Parameter:status:(success: 0x00/failure: others)

5.17. writedReceived

- (void)writedReceived:(uint8_t)statusCode;

Description: This function is called when a response code to "Write to TagMemory" is received.
Parameter:statusCode:(success: 0x00/failure: others)

5.18. stoppedReadTags

- (void)stoppedReadTags:(uint8_t)statusCode;

Description: This function is called when tag reading is stopped.
Parameter:statusCode:(success:0x00/failure: others)

5.19. lockedReceived

- (void)lockedReceived:(uint8_t)statusCode;

Description: This function is called when a response code to "Lock Type C Tag" is received.
Parameter:statusCode: (success: 0x00/failure: others)

5.20. didSetFhLbtReceived

- (void)didSetFhLbtReceived:(uint8_t)statusCode;

Description: This function is called when a response code to "Set FhLbt" is received.
Parameter: status:(success:0x00/failure: others)

5.21. didSetAntiColModeReceived

- (void)didSetAntiColModeReceived:(uint8_t)statusCode;

Description: This function is called when a response code to "Set Anti-Collision Mode" is received.
Parameter: statusCode:(success:0x00/failure: others)

5.22. sessionReceived

- (void)sessionReceived:(uint8_t)session;

Description: This function is called when a response code to "Set Session" is received.
Parameter: session : S0(0x00), S1(0x01), S2(0x02), S3(0x03), Dev.mode(0xF0)

5.23. didSetStopConditionMtnu

- (void)didSetStopConditionMtnu:(uint8_t)statusCode;

Description: This function is called when a response code to "Set Stop Condition" is received.
Parameter:statusCode:(success:0x00/failure: others)

5.24. didSetOptiFreqHPTable

- (void)didSetOptiFreqHPTable:(uint8_t)statusCode;

Description: This function is called when a response code to "Set FH and LBT Parameters" is received.
Parameter:statusCode:(success:0x00/failure: others)

5.25. didSetFreqHPMode

- (void)didSetFreqHPMode:(uint8_t)statusCode;

Description: This function is called when a response code to "Set FreqHopping Table" is received.
Parameter:statusCode:(success:0x00/failure:others)

5.26. didSetSession

- (void)didSetSession:(uint8_t)statusCode;

Description: This function is called when a response code to "Set Session" is received.
Parameter: statusCode:(success:0x00/failure:others)

5.27. txPowerLevelReceived

- (void)txPowerLevelReceived:(NSData*)power;

Description:

Response of "getOutputPowerLevel". Assign the RFID TX Power value to the object of the commonReadInfo class after the response.
fRFIDpower: current output power
fRFIDpowerMax: Maximum output power that can be set
fRFIDpowerMin: Minimum output power that can be set

5.28. pcEpcSensorDataReceived

– (void)pcEpcSensorDataReceived:(NSData *)pcEpc sensorData:(NSData *)sensorData;

Description: This function is called when tag data with sensor data is received.
Parameter:pcEpc: pcepc data
sensorValue: sensor value
Sample code:

```
- (void)pcEpcSensorDataReceived:(NSData *)pcEpc sensorData:(NSData
*)sensorData
{
    int codeType;// Tag type, 2 (humidity tag) / 3 (temperature tag)
    int onChipRssiCodeValue;// Tag chips RSSI data
    int sensorCodeValue;// temperature / humidity data (hexadecimal)
    double calcTemp;// temperature (Celsius)
    NSMutableString *tmptagid;// tag pcepc data (hexadecimal number)

    NSData *tagid = pcEpc;
    NSData *taghex = sensorData;

    //cepc NSData transformation NSString
    tmptagid = [[NSMutableString alloc] init];
    unsigned char* ptrtagid= (unsigned char*) [tagid bytes];
    for(int i = 0; i < tagid.length; i++)
        [tmptagid appendFormat:@"%02X", *ptrtagid++ & 0xFF ];

    //Temperature / humidity data analysis
    Byte *b = (Byte*) [taghex bytes];
    codeType = b[0];
    onChipRssiCodeValue = (b[1] << 8) | b[2];
    sensorCodeValue = (b[3] << 8) | b[4];
    double code1 = 0;
    double temp1 = 0;
    double code2 = 0;
    double temp2 = 0;
    double tempCode = sensorCodeValue;
    if (codeType == 3) {
        int temp = b[7] << 4;
        code1 = temp + ((b[8] >> 4) & 0x0F);
```

```
            temp = (b[8] & 0x0F) << 7;
            temp1 = temp + ((b[9] >> 1) & 0x7F);
            temp = (b[9] & 0x01) << 8;
            temp = (temp + b[10]) << 3;
            code2 = temp + ((b[11] >> 5) & 0x07);
            temp = (b[11] & 0x1F) << 6;
            temp2 = temp + ((b[12] >> 2) & 0x3F);
            calcTemp = ((temp2 - temp1) / (code2 - code1) * (tempCode - code1) +
temp1 - 800) / 10;
      }
}
```