AsReader Combo SDK Manual

AsReader Combo SDK Manual V1.1

For ASR-0230D, ASR-0231D, ASR-0240D

Asterisk,Inc.
2018-07-19

Modification

No.	Version	Modified Content	Date
1	1.1	Initial version	2018/07/19

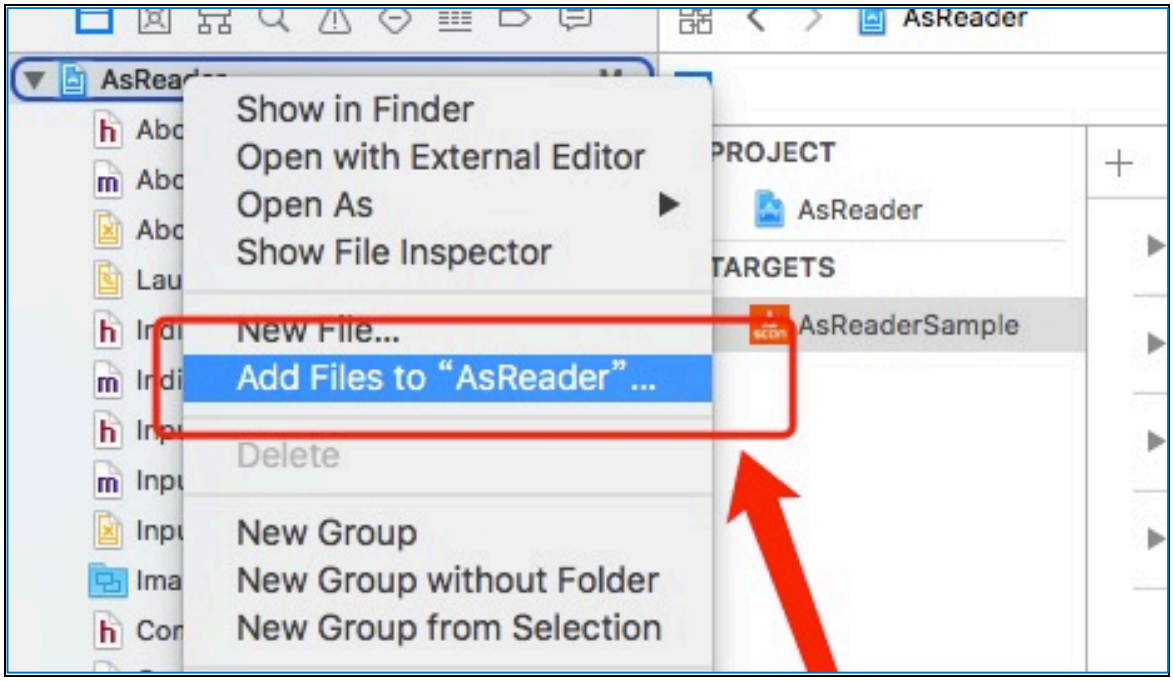
Contents

1. SDK Usage

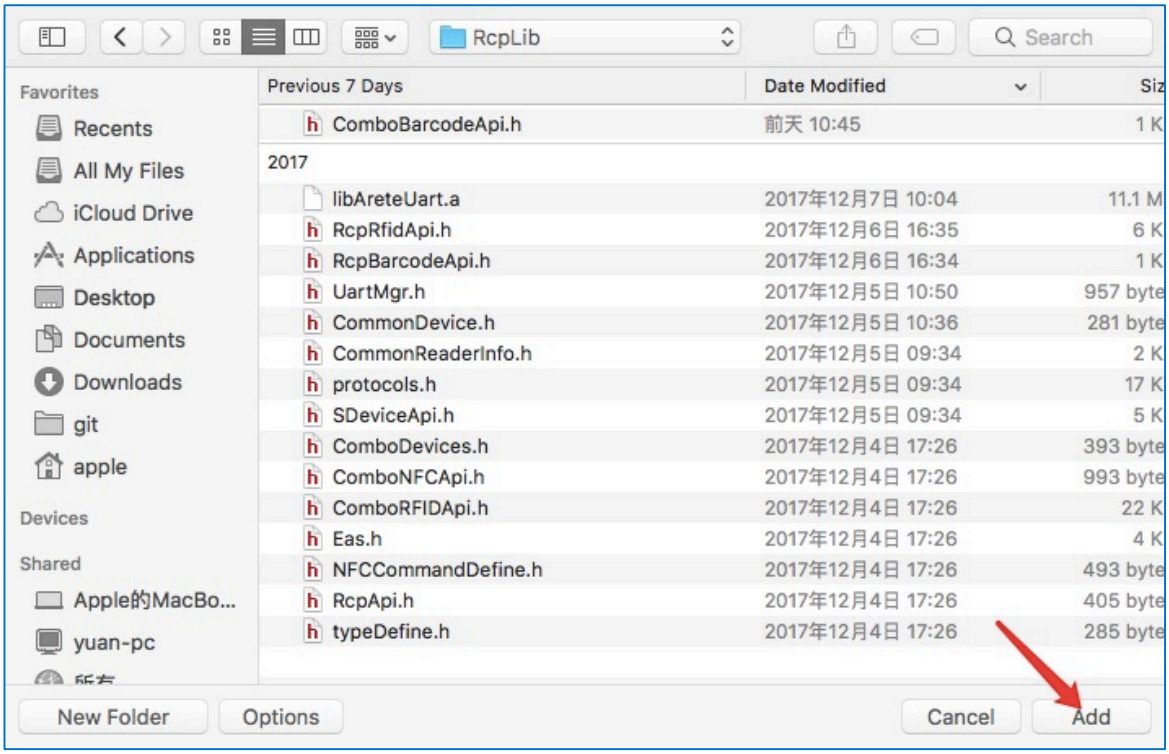
1.1. Import header files of the SDK to project

- #import “typeDefine.h”
- #import “protocols.h”
- #import “SDeviceApi.h”
- #import “RcpApi.h”
- #import “ComboNFCApi.h”
- #import “ComboRFIDApi.h”
- #import “ComboBarcodeApi.h”
- #import “CommonDevice.h”
- #import “CommonReaderInfo.h”
- #import “NFCCommandDefine.h”
- #import “ComboDevices.h”

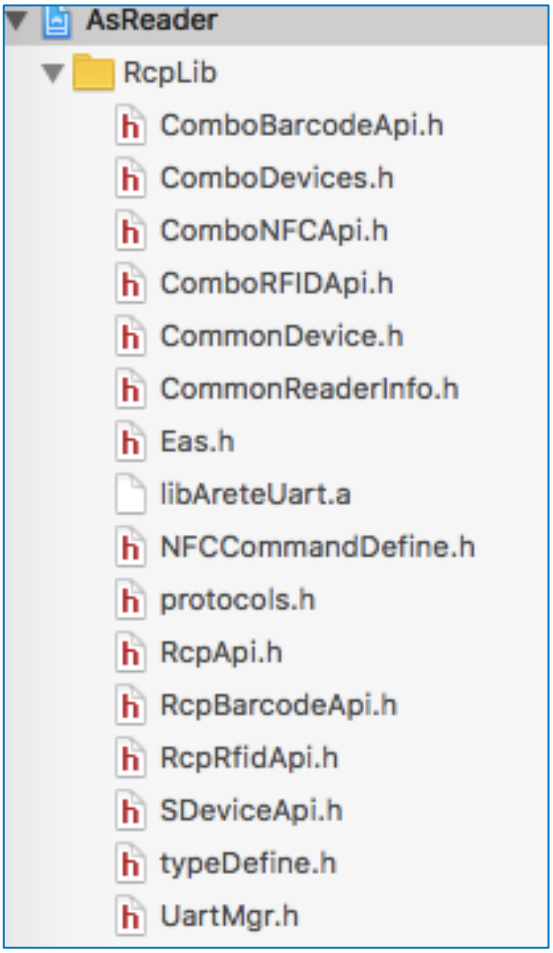
1.1.1 “Add Files to”



1.1.2 Select “Add”

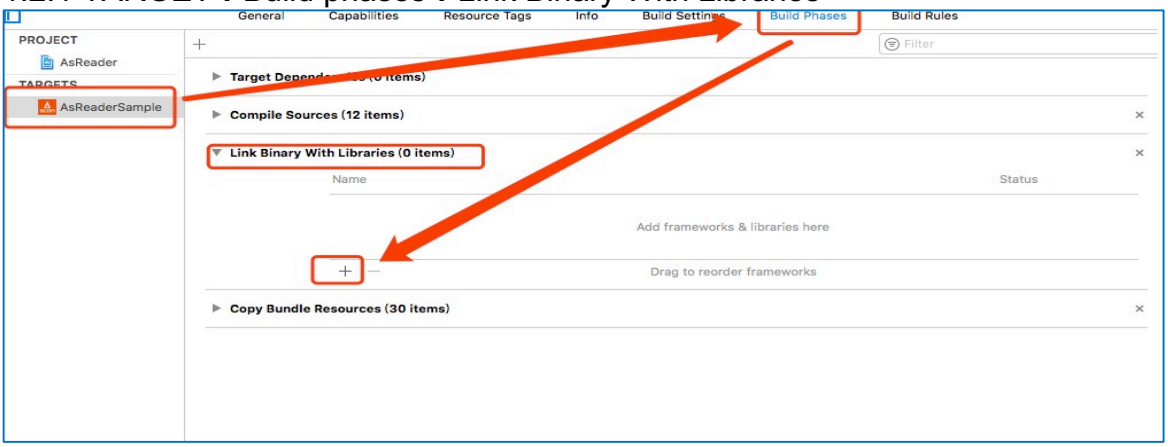


1.1.3 Complete as shown

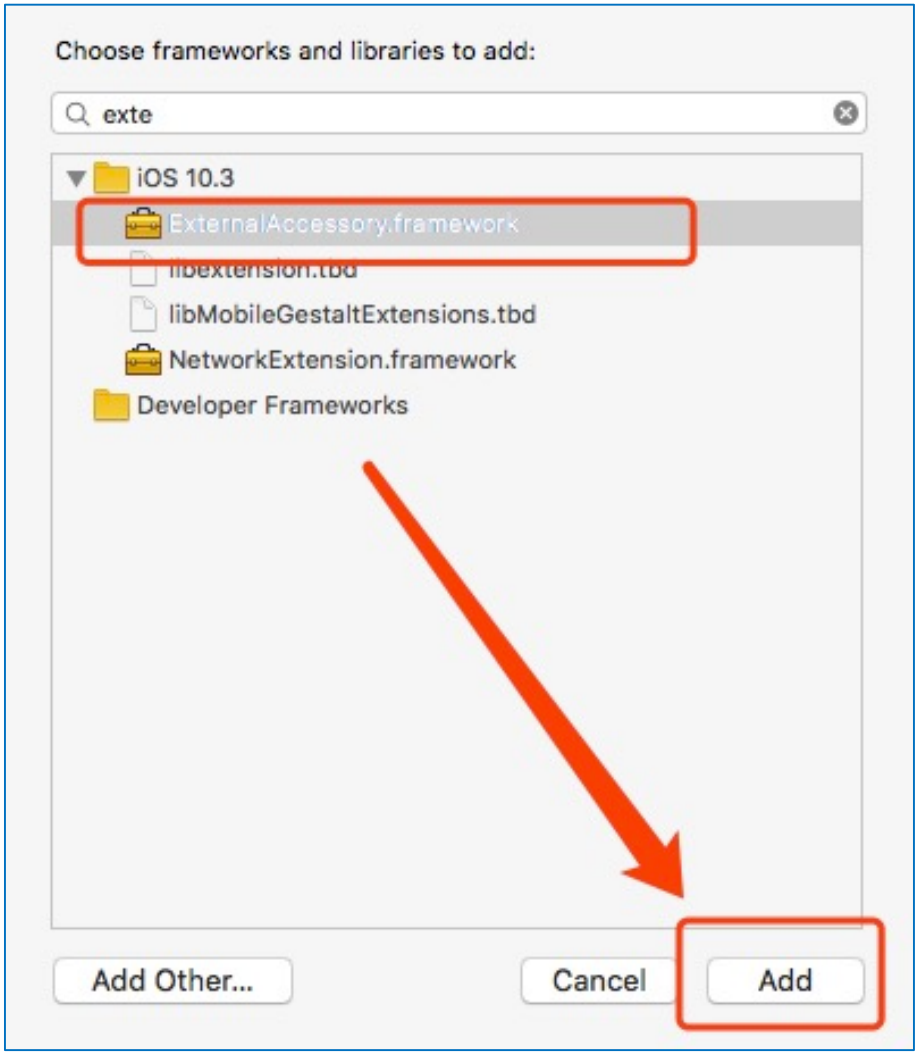


1.2. Add ExternalAccessory.Framework

1.2.1 TARGET→Build phases→Link Binary With Libraries

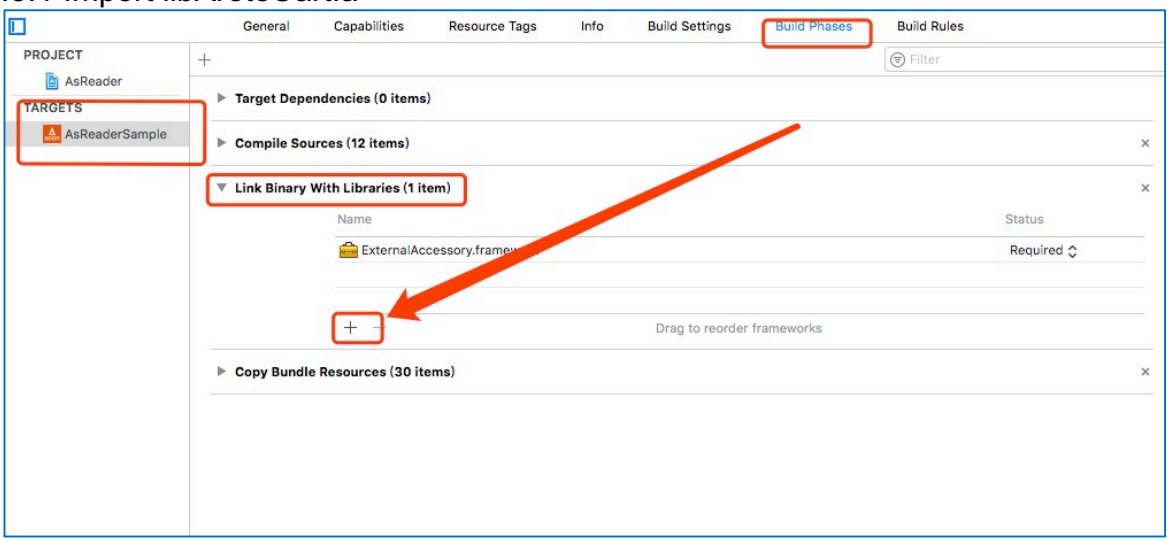


1.2.2 Select “Add”

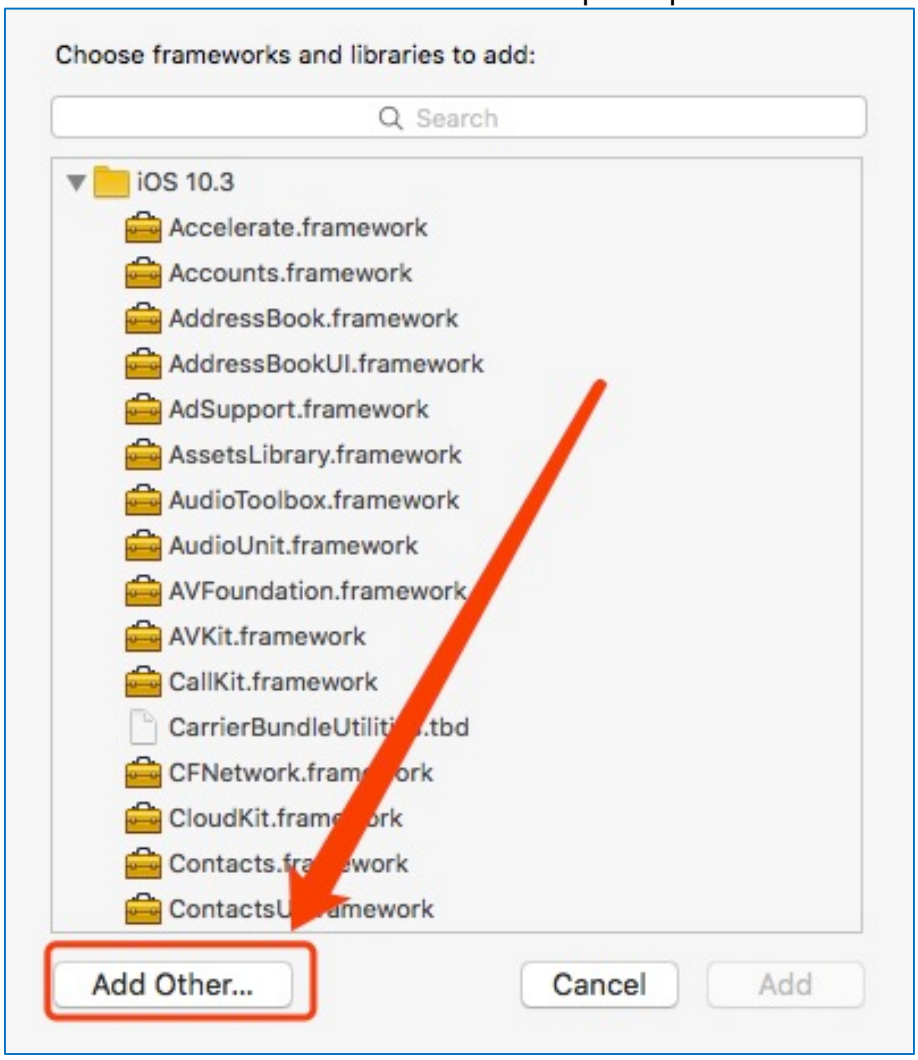


1.3. Import libAreteUart.a

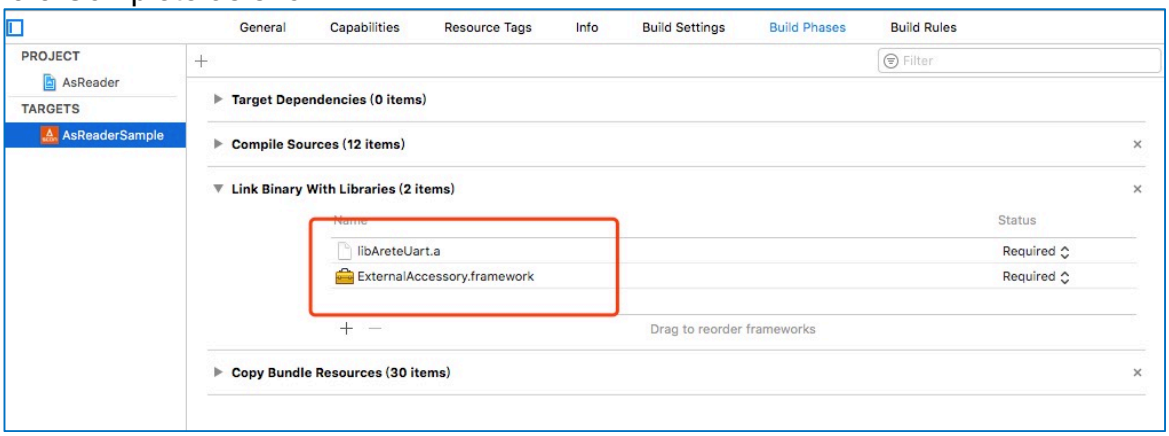
1.3.1 Import libAreteUart.a



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



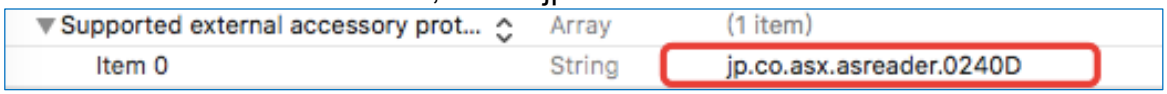
1.3.3 Complete as shown



1.4. Add AsReader protocol

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

ASR-0230D,0240D:jp.co.asx.asreader.0240D



1.5. Use SDK In Class

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

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. Description of Methods

2.1. ComboBarcodeApi

Supported AsReader: ASX-510R,ASX-520R,ASR-010D,ASR-020D,ASR-0230D,ASR-0231D.

2.1.1. sharedInstance

+ (id)sharedInstance;

Description: Create a ComboBarcodeApi object (Singleton Pattern).
Return Value: ComboBarcodeApi object

2.1.2. startScan

- (BOOL)startScan;

Description: The reader starts scanning barcodes.
Return Value:
YES: success
NO: failure

2.1.3. stopScan

- (BOOL)stopScan;

Description: The reader stops scanning barcodes.
Return Value:
YES: success
NO: failure

2.1.4. setFactoryReset

Note: this method only supports the barcode mode for ASR-0230D, ASR-0231D, and ASR-0240D.

-(BOOL)setFactoryReset;

Description: Factory reset (barcode mode).
Return Value:
YES: success
NO: failure

2.1.5. setSymbologyPrefix

Note: this method only supports the barcode mode for ASR-0230D, ASR-0231D, and ASR-0240D.

-(BOOL)setSymbologyPrefix;

Description: Display or hide the prefix of a barcode. (The read result for a barcode “123”, when display prefix is enabled, will be “A123”).Display or hide prefix of barcode. (barcode “123”,when set to display prefix, the read result will be “A123”).
Return Value:
YES: success.
NO: the device does not support this feature or current mode is not barcode scanning mode.

2.2. ComboNFCApi Class

Supported AsReader:ASR-0240D

2.2.1. sharedInstance

+ (id) sharedInstance;

Description: Create a ComboNFCApi object (Singleton Pattern).
Return Value: ComboNFCApi object

2.2.2. sendRawData

- (BOOL) sendRawData:(Byte*)sendData len:(int)nLen;

Description: Send raw data to the reader.
Parameter:
sendData: send data
nLen:sendData:data length (byte)
Return Value:
YES: success
NO: failure

2.2.3. startScan

- (BOOL) startScan;

Description:NFC type reader starts to scan tags.
Return Value:
YES: success
NO: failure

2.2.4. stopScan

- (BOOL) stopScan;

Description:NFC type reader stops scanning tags.
Return Value:
YES: success
NO: failure

2.3. ComboRFIDApi

2.3.1. sharedInstance

```
+ (id)   sharedInstance;
```

Description: Create a ComboRFIDApi object (Singleton Pattern).
Return Value: ComboRFIDApi object

2.3.2. startScan

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

Description: Start an automatic tag (RFID) read operation, tag IDs are sent back to user through notification packets.
Parameter:
mtnu:Maximum number of tag 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

2.3.3. stopScan

```
- (BOOL)stopScan;
```

Description: Stop reading tags.
Return Value:
YES: success
NO: failure

2.3.4. startReadTagsWithRssi:

```
- (BOOL)startReadTagsWithRssi:(uint8_t)maxTags
mtime:(uint8_t)maxTime repeatCycle:(uint16_t)repeatCycle;
```

Description: Start reading tags with RSSI.
Parameter:
maxTags:Maximum number of tags to read
maxTime:Maximum elapsed time to read tags (sec)
repeatCycle:How many times the reader performs an inventory round
Return Value:
YES: success
NO: failure

2.3.5. 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

2.3.6. setOnOffTimeSetOnTime

```
- (BOOL)setOnOffTimeSetOnTime:(uint16_t)ReadTime
                    setOffTeim:(uint16_t)IdleTime;
```

Description: Get the RF channel. This command is valid only for non-FH mode. Set only ReadTime and IdleTime in FH and LBT Parameters. continuousWave is continuousWave 0.

Parameter:

ReadTime:read time(ms)
IdleTime:idle time(ms)

Return Value:

YES: success
NO: failure

2.3.7. setChannel

```
- (BOOL)setChannel:(uint8_t)channel
     channelOffset:(unit8_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

2.3.8. 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

2.3.9. getOutputPowerLevel

```
- (BOOL)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

2.3.10. setOutputPowerLevel

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

Description: To set the current Tx power.
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

2.3.11. 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 0x00000000
epc:target data
memoryBank: reserved: 0x00/EPC:0x01/TID: 0x02/User:0x03
startAddress: starting address word pointer
dataToWrite:data to write

Return Value:

YES: success
NO: failure

2.3.12. killTag

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

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

killPassword: kill password. If KP is set to 0x00000000, the ‘Kill Type C Tag’ command will not work. The target tag will ignore it.

Return Value:

YES: success
NO: failure

2.3.13. 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 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

2.3.14. getFreqHoppingTable

- (BOOL)getFreqHoppingTable;

Description: Send the "Get Frequency Hopping Table" command to the reader to get the current frequency hopping table.

Return Value:

YES: success

NO: failure

2.3.15. getSession

- (BOOL)getSession;

Description: Send the "Get Session" command to the reader to get the current session.

Return Value:

YES: success

NO: failure

2.3.16. 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

2.3.17. 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

2.3.18. setAnticollision

```
- (BOOL)setAnticollision:(uint8_t)mode
                 Counter:(uint8_t)counter;
```

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

counter:change target at N-th Tx On according to inventory round result (default:1)

Return Value:

YES: success

NO: failure

2.3.19. updateRegistry

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

Description: Update registry.

Return Value:

YES: success

NO: failure

2.3.20. getRFIDOnOffTime;

```
- (BOOL)getRFIDOnOffTime;
```

Description: Call getFhLbtParam (self getFhLbtParam).

Return Value:

YES:success

NO:failure

2.3.21. getRFIDModuleVersion

```
- (BOOL)getRFIDMoudleVersion;
```

Description: Send the "Get Reader Information" command to the reader to get basic information from the reader.

Return Value:

YES: success

NO: failure

2.3.22. setHoppingOnOff

```
- (BOOL)setHoppingOnOff:(BOOL)isOn;
```

Description: Send the "Set FH and LBT Parameters" command to the reader. Only set frequencyHopping and listenBeforeTalk in FH and LBT Parameters. continuousWave is continuousWave 0.

Parameter: isOn:

YES:Set frequencyHopping is 2 and listenBeforeTalk is 1.

NO:Set frequencyHopping is 1 and listenBeforeTalk is 2.

Return Value:

YES: success

NO: failure

2.3.23. writeToTagMemoryWithEPC

```
- (BOOL)writeToTagMemoryWithEPC:(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: success
NO: failure

2.3.24. readFromTagMemory

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

Description: Read indicated Type C tag memory.

Parameter:

accessPassword:access password
epc:target tag's epc
memoryBank:RFU (0x00), EPC (0x01), TID (0x02), User (0x03)
startAddress:starting address
dataLength: data length

Return Value:

YES: success
NO: failure

2.3.25. setOptimumFrequencyHoppingTable

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

Description: Set optimum frequency hopping table.

Return Value:

YES: success
NO: failure

2.3.26. getFrequencyHoppingMode

```
- (BOOL) getFrequencyHoppingMode;
```

Description: Send the "Get Frequency Hopping Mode" command to the reader to get frequency hopping mode.

Return Value:

YES: success
NO: failure

2.3.27. getStopCondition

```
- (BOOL) getStopCondition;
```

Description: Send the "Get Stop Condition" command to the reader to get the stop point of start-auto-read.

Return Value:

YES: success
NO: failure

2.3.28. setSmartHoppingOnOff

```
- (BOOL)setSmartHoppingOnOff:(BOOL)isOn;
```

Description: Send the "Set Frequency Hopping Mode" command to the reader to set frequency hopping mode.

Parameter:isOn:FH mode; YES: smart hopping mode/NO: normal mode

Return Value:

YES: success
NO: failure

2.4. CommonDevice

Supported AsReader: ASX-300R,ASX-301R,ASX-510R,ASX-520R,ASR-010D,ASR-020D,ASR-030D,ASR-031D,ASR-0230D,ASR-0231D,ASR-0240D.

2.4.1. showPrintNSLogadc

```
+ (void)showPrintNSLog:(BOOL)isShow;
```

Description: Print log (used when debugging to output the log to the console)
Parameter:isShow:
YES: show log
NO: do not show log

2.4.2. setTriggerModeDefault

Note: this method only supports Barcode mode for ASR-0230D, ASR-0231D, and ASR-0240D.

```
+(void)setTriggerModeDefault:(BOOL)isDefault;
```

Description: Set AsReader trigger default mode.
Parameter:
YES:execute default trigger mode (scan)
NO:custom mode

2.5. HWEventDelegate

Supported AsReader: ASX-300R,ASX-301R,ASX-510R,ASX-520R,ASR-010D,ASR-020D,ASR-030D,ASR-031D,ASR-0230D,ASR-0231D,ASR-0240D.

2.5.1. resPowerOnOff

```
- (void)resPowerOnOff:(BOOL)isOn
               Device:(int)nDeviceType
      IsHWModeChange:(BOOL)bIsHWModeChange;
```

Description: This function is called when the reader sends a response code to "setReaderPower".

Parameter:

isOn:On:YES/Off:NO
nDeviceType:Unknown:99/Barcode:0/RFID:1/NFC:2
bIsHWModeChange:YES:When mode is changed by H/W switch

2.5.2. readerConnected

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

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

Parameter:status:connected:0xFF/disconnected:0x00.

2.5.3. checkTriggerStatus

```
- (void)checkTriggerStatus:(NSString*)strStatus;
```

Description: This function is called when the trigger button of the reader is pressed or released.

Parameter:strStatus

When device type is RFID, "RFID startScan"
When device type is Barcode, "Barcode startScan"
When device type is NFC, "NFC startScan"

2.5.4. plugged

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

Description: Tis function is called when the plug state between the reader and iPhone changes.

Parameter: plug:plugged: YES/unplugged: NO

2.5.5. pushedTriggerButton

```
- (void)pushedTriggerButton;
```

Description: The results of the change will be called back when pressing the trigger button on the AsReader.

2.5.6. releasedTriggerButton

```
- (void)releasedTriggerButton;
```

Description: The results of the change will be called back when stopping pressing the trigger button on the AsReader.

2.6. RcpCommonDelegate

Supported AsReader: ASX-300R,ASX-301R,ASX-510R,ASX-520R,ASR-010D,ASR-020D,ASR-030D,ASR-031D,ASR-0230D,ASR-0231D,ASR-0240D.

2.6.1. errReceived

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

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

2.6.2. adcReceived

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

Description: This function is called when the battery level of reader is received.
Parameter: data: battery level

2.6.3. receivedScanData

```
- (void)receivedScanData:(NSData *)readData
                DeviceType:(int)nDeviceType;
```

Description: This function is called when tag data is received.
Parameter:
readData: tag data
nDeviceType: unknown: 99/barcode:0 /RFID:1/NFC:2

2.6.4. resFactoryRse

Note: this method only supports Barcode mode for ASR-0230D, ASR-0231D, and ASR-0240D.

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

Description: Response of "Barcode Factory Reset". This function is called when the reader sends a response code to "Barcode Factory Reset".
Parameter: status: reset start:0x00/reset complete:0xFF

2.6.5. nfcRawDataReceived

```
- (void)nfcRawDataReceived:(NSData *)rawData;
```

Description: This function is called when tag data (nfc type) is received.
Parameter:rawData: nfc tag data

2.6.6. allRawDataReceived

```
- (void)allRawDataReceived:(NSData *)rawData;
```

Description: This function is called when tag data (all types) is received.
Parameter: rawData: tag data

2.7. RcpRFIDDelegate

Supported AsReader: ASX-300R,ASX-301R,ASR-030D,ASR-031D,ASR-0230D,ASR-0231D.

2.7.1. epcReceived

```
- (void)epcReceived:(NSData *)epc tid:(NSData *)tid;
```

Description: This function is called when tag data with TID is received.
Parameter:
epc:epc data
tid:tid data

2.7.2. rssiReceive

```
- (void)rssiReceived:(uint16_t)rssi;
```

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

2.7.3. didSetOutputPowerLevel

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

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

2.7.4. didSetChParamReceived

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

Description: Response of "Set current RF Channel". This function is called when a response code to "Set current RF Channel" is received.
Parameter:statusCode:success: 0x00/failure: others

2.7.5. didSetAntiCol

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

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

2.7.6. didSetSession

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

Description: Response of "Set Session". This function is called when a response code to "Set Session" is received.
Parameter: status:success:0x00/failure:others

2.7.7. channelReceived

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

Description: Response of "Get current RF Channel". 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

2.7.8. anticolParamReceived

- **(void)anticolParamReceived:(uint8_t)mode Counter:(uint8_t)counter;**

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

Parameter:

mode:fixed Q: 0x00/dynamic Q:0x01

counter :counter value

2.7.9. 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

2.7.10. regionReceived

- (void)regionReceived:(uint8_t)region;

Description: Response of "Get Region". 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)

2.7.11. onOffTimeChanged

- (void)onOffTimeChanged;

Description: Response of "Set FH and LBT Parameters". This function is called when a response code to "Set FH and LBT Parameters" is received.

2.7.12. fhLbtReceived

- (void)f'hLbtReceived:(NSData *)fhLb;

Description: Response of "getFhLbtParam".

Parameter: fhLb: Read time (16 bits), idle time (16 bits), carrier monitoring time (16 bits), target RF power level (16 bits), FH (8 bits),LBT (8 bits), CW (8 bits)

2.7.13. hoppingTableReceived

- (void)hoppingTableReceived:(NSData *)table;

Description: Response of "Get Frequency Hopping Table". This function is called when a response code to "Get Frequency Hopping Table" is received.

Parameter:table: table size (8bit)

2.7.14. didSetFHLbt

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

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

2.7.15. didSetOptiFreqHPTable

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

Description: Response of "Set Optimum Frequency Hopping Table". This function is called when a response code to "Set Optimum Frequency Hopping Table" is received.
Parameter:status:start:0x00/finish: 0x01

2.7.16. didSetFHmodeChanged

```
- (void)didSetFHmodeChanged;
```

Description: Response of "Set Frequency Hopping Mode". This function is called when a response code to "Set Frequency Hopping Mode" is received.

2.7.17. resGetFHmode

```
- (void)resGetFHmode;
```

Description: Response of "Get Frequency Hopping Mode". This function is called when a response code to "Get Frequency Hopping Mode" is received.

2.7.18. rfidModuleVersionReceived

```
- (void)rfidModuleVersionReceived;
```

Description: Response of "Get Reader Information". This function is called when a response code to "Get Reader Information" is received.

2.7.19. rfidOnOffTimeReceived

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

Description: Response of "Get FH and LBT Parameters". This function is called when a response code to "Get FH and LBT Parameters" is received.

2.7.20. writedReceived

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

Description: Response of "Write Type C Tag Data". This function is called when a response code to "Write Type C Tag Data" is received.
Parameter: statusCode: success:0x00/failure: others

2.7.21. sessionReceived

```
- (void)sessionReceived:(uint8_t)session;
```

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

2.7.22. tagMemoryReceived

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

Description: Response of "readFromTagMemory".
Parameter:data:Return the data of the area.

2.7.23. killedReceived

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

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

2.7.24. lockedReceived

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

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

2.7.25. responseReboot

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

Description: This function is called when rebooting (firmware update).
Parameter:status:success: 0x00/failure: others

2.7.26. updatedRegistry

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

Description: Response of "Update Registry". This function is called when a response code to "Update Registry" is received.
Parameter:status:success: 0x00/failure: others

2.8. CommonReaderInfo

You can access the device information via the class CommonReaderInfo.

```
@property(nonatomic,readonly) NSString *strName;                    //device name
@property(nonatomic,readonly) NSString *strfirmware;                //device F/W
@property(nonatomic,readonly) NSString *strhardware;                //device H/W
@property(nonatomic,readonly) NSString *strID;                      //device ID
@property(nonatomic,readonly) NSString *strmanufacturer;            //manufacturer
@property(nonatomic,readonly) NSString *strmodelNumber;             //model number
@property(nonatomic,readonly) NSString *strserialNumber;            //serial number
@property(nonatomic,readonly) NSString *strProtocol;                //protocol
@property(readonly,assign) int m_nReaderType;                       //reader type
@property(readonly,assign) int m_nCurrentSelectDevice;              //selected device
@property(readonly,assign) BOOL bCanUseRFID;                        //usable RFID
@property(readonly,assign) BOOL bCanUseBarcode;                     //usable Barcode
@property(readonly,assign) BOOL bCanUseNFC;                         //usable NFC
@property(readonly,assign) BOOL bIsPowerOn;                         //power on
@property(readonly,assign) BOOL bBeep;                              //beep
@property(readonly,assign) BOOL bVirbration;                        //vibration
@property(readonly,assign) BOOL bLED;                               //LED
@property(readonly,assign) BOOL bIllumination;                      //illumination
@property(readonly,assign) BOOL bSymbologyPrefix;                   //symbologyprefix
@property(readonly,assign) BOOL bIsTriggerModeDefault;              //default trigger mode
@property(readonly,assign) float fRFIDpower;                        //RFID power
@property(readonly,assign) float fRFIDpowerMax;                     //RFID max power
@property(readonly,assign) float fRFIDpowerMin;                     //RFID min power
@property(readonly,assign) int     nRFIDonTime;                     //RFID on time
@property(readonly,assign) int     nRFIDoffTime;                    //RFID off time
@property(readonly,assign) int     nRFIDchannel;                    //RFID channel
@property(readonly,assign) int     nCount;                          //tag count
@property(readonly,assign) int     nScanTime;                       //scan time
@property(readonly,assign) int     nCycle;                          //scan cycle
@property(readonly,assign) int     nCst;                            //carriersense time
@property(readonly,assign) int     nRfl;                            //RF level
@property(readonly,assign) int     nLbt;                            //RFID LBT
@property(readonly,assign) int     nFh;                             //Frequency Hopping
@property(readonly,assign) int     nCw;                             //continuous wave
@property(readonly,assign) BOOL bSmartHopping;                      //smart hopping
@property(nonatomic,readonly) NSString *strRFIDModuleVersion; //RFID version
```