AsReader DOCK SDK

SDK マニュアル V1.2

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

Asterisk, Inc.
2018-06-11

変更履歴

序号	バージョン	変更内容	日付
1	1.0	新規作成	2018/03/22
2	1.1	温度/湿度の追加方法 startReadTagsRFM/pcEpcSensorDataReceived	2018/05/28
3	1.2	getOutputPowerLevel/setOutputPowerLevel/ txPowerLevelReceived メソッドの説明の修正	2018/06/11

目次

1 SDK を使用するために 4
1.1 使用する SDK のヘッダーファイルのインポートします。 4
1.2 ExternalAccessory.framework の追加 5
1.3 libAreteUart.a のインポート 6
1.4 AsReader protocol の追加 7
1.5 SDK のインポート 8
1.6 注意事項 8
2 BarcodeApi Class 9
2.1 init 9
2.2 open 9
2.3 isOpened 9
2.4 close 9
2.5 startReadBarcodes 9
2.6 setReaderPower 10
2.7 setBeep 10
2.8 getSDKVersion 10
3 BarcodeDelegate Class 11
3.1 readerConnected 11
3.2 pluggedBarcode 11
3.3 barcodeStringReceived 11
3.4 batteryChargeReceived 11
4 RfidApi Class 12
4.1 getSDKVersion 12
4.2 init 12
4.3 open 12
4.4 isOpened 12
4.5 close 12
4.6 setReaderPower 13
4.7 setBeep 13
4.8 startReadTags 13
4.9 startReadTagsWithRssi 14
4.10 stopReadTags 14
4.11 getChannel 14
4.12 setChannel 14
4.13 getFhLbtParam 15
4.14 setFhLbtParam 15
4.15 getOutputPowerLevel 15
4.16 setOutputPowerLevel 16
4.17 readFromTagMemory 16
4.18 getSession 16
4.19 setSession 16
4.20 getAnticollision 17
4.21 setAnticollision 17
4.22 writeToTagMemory 17
4.23 killTag 18
4.24 lockTagMemory 18
4.25 setStopConditionMtnu 18

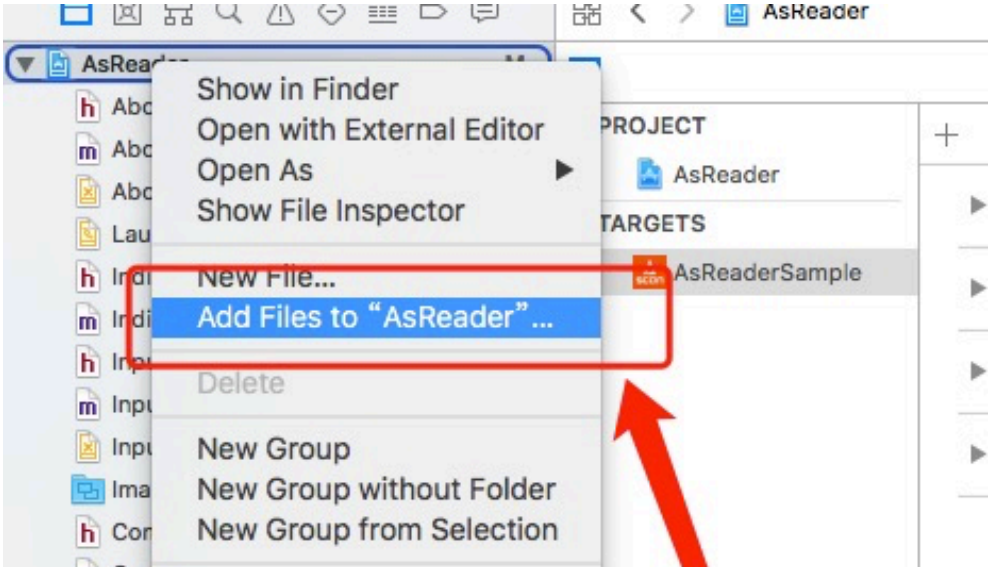
4.26 setOptimumFrequencyHoppingTable 18
4.27 SetFrequencyHoppingMode 19
4.28 updateRegistry 19
4.29 writeToTagMemory 19
4.30 startReadTagsRFM 19
5 RfidDelegate Class 20
5.1 pluggedRfid 20
5.2 pcEpcReceived 20
5.3 pcEpcRssiReceived 20
5.4 readerConnected 20
5.5 readerConnected 20
5.6 errReceived 21
5.7 errDetailReceived 21
5.8 frequencyHoppingModeReceived 23
5.9 regionReceived 23
5.10 channelReceived 23
5.11 fhLbtReceived 23
5.12 tagMemoryReceived 23
5.13 anticolParamReceived 23
5.14 batteryChargeReceived 24
5.15 startedReadTags 24
5.16 didSetOutputPowerLevel 24
5.17 writedReceived 24
5.18 stoppedReadTags 25
5.19 lockedReceived 25
5.20 didSetFhLbtReceived 25
5.21 didSetAntiColModeReceived 25
5.22 sessionReceived 25
5.23 didSetStopConditionMtnu 25
5.24 didSetOptiFreqHPTable 26
5.25 didSetFreqHPMode 26
5.26 didSetSession 26
5.27 txPowerLevelReceived 26
5.28 pcEpcSensorDataReceived 26

1 SDK を使用するために

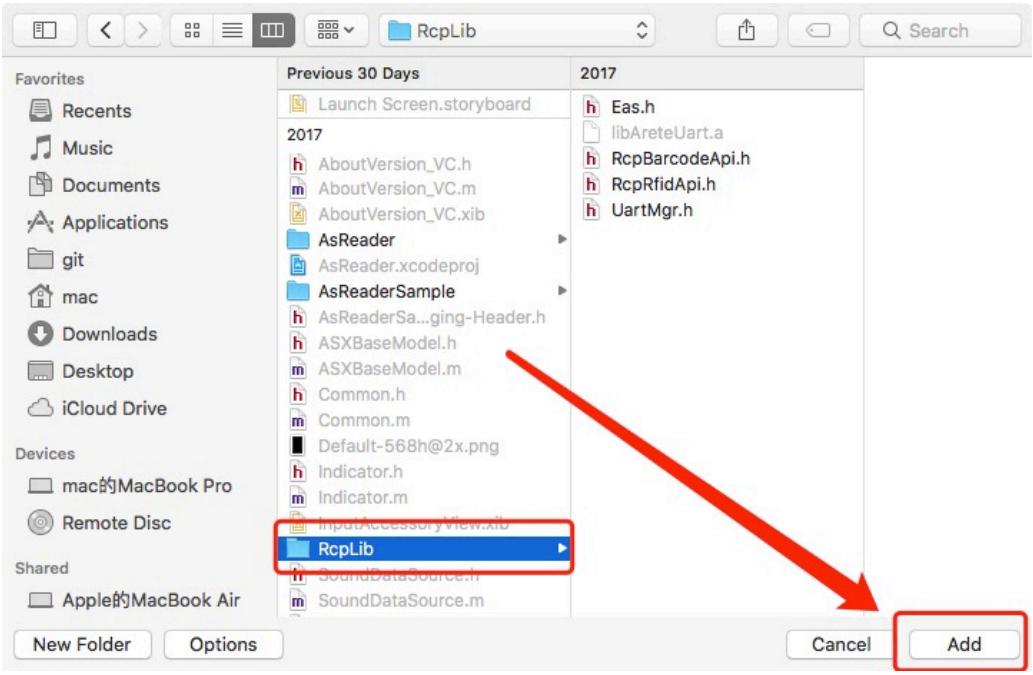
1.1 使用する SDK のヘッダーファイルのインポートします。

- #import “Eash.h”
- #import “UarMgr.h”
- #import “RcpBarcodeApi.h”
- #import “RcpRfidApi.h”

1.1.1 「Add Files to」を選択します。



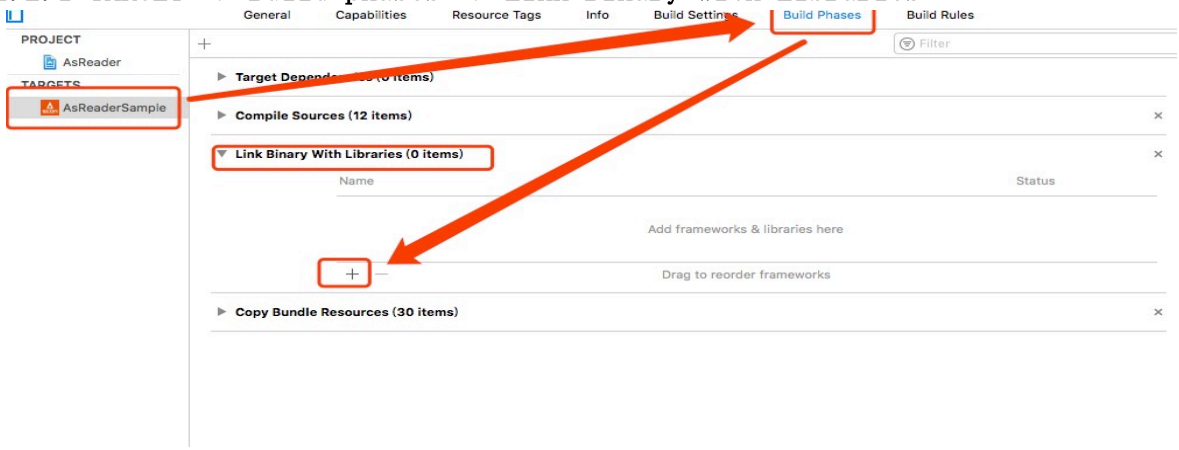
1.1.2 SDK を選択し、「Add」 します。



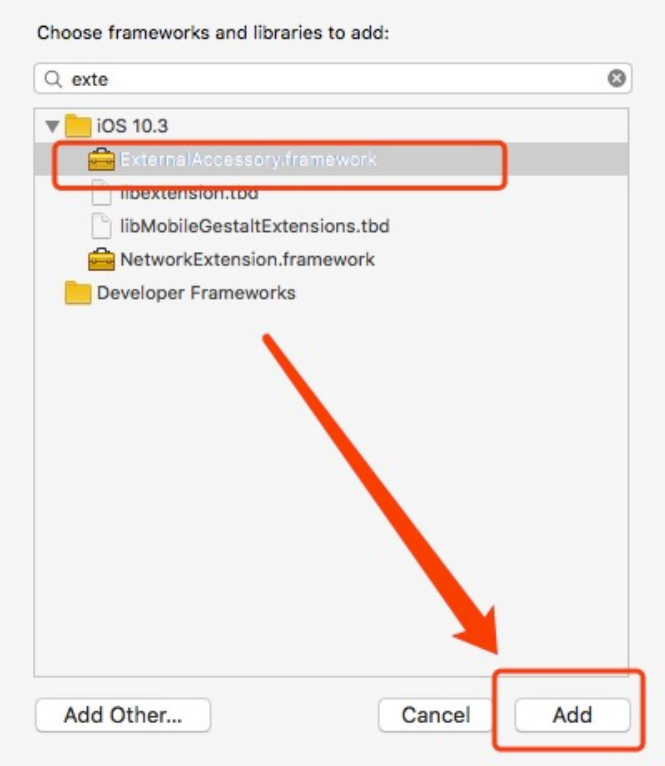
1.1.3 SDK が追加されていることを確認します。

1.2 ExternalAccessory.framework の追加

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



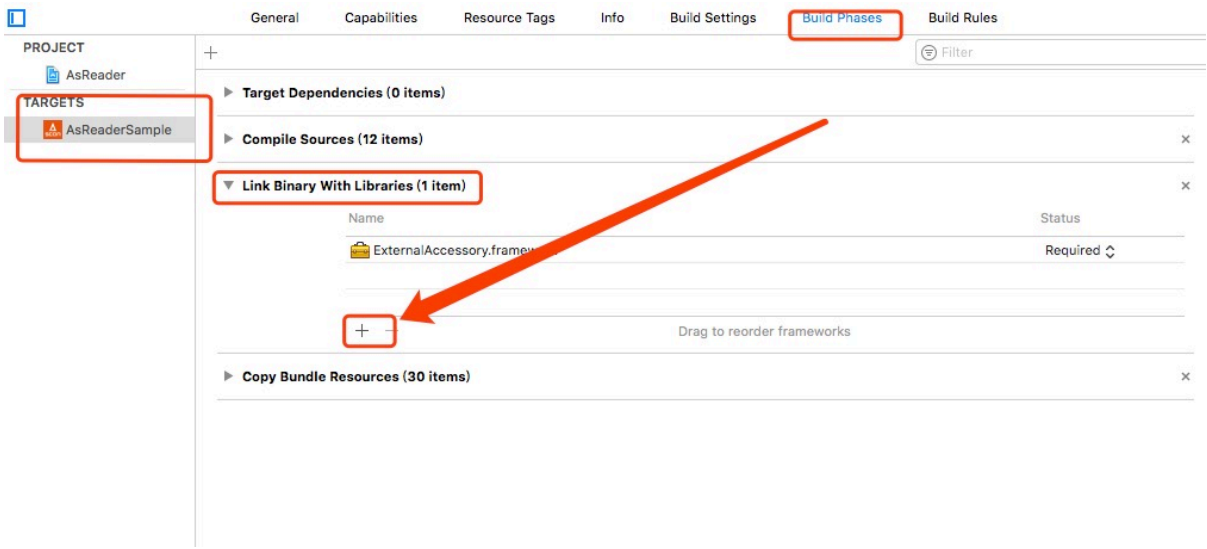
1.2.2 「ExternalAccessory.framework」を選択し、「Add」 をクリックします。



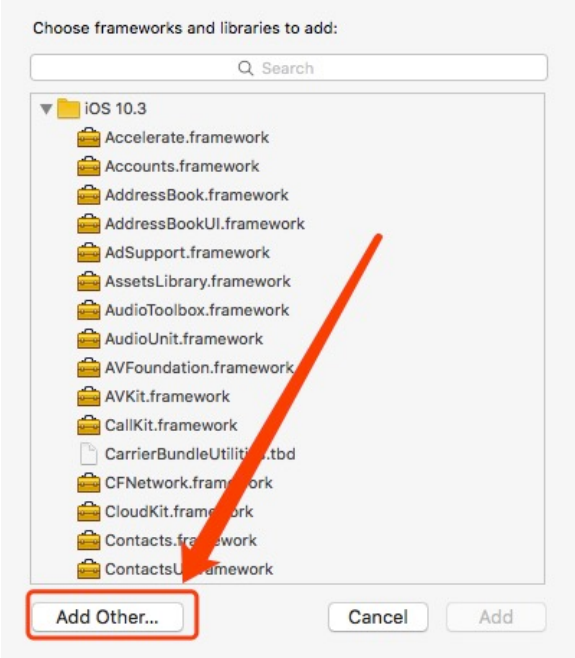
1.2.3 「ExternalAccessory.framework」が追加されていることを確認します。

1.3 libAreteUart.a のインポート

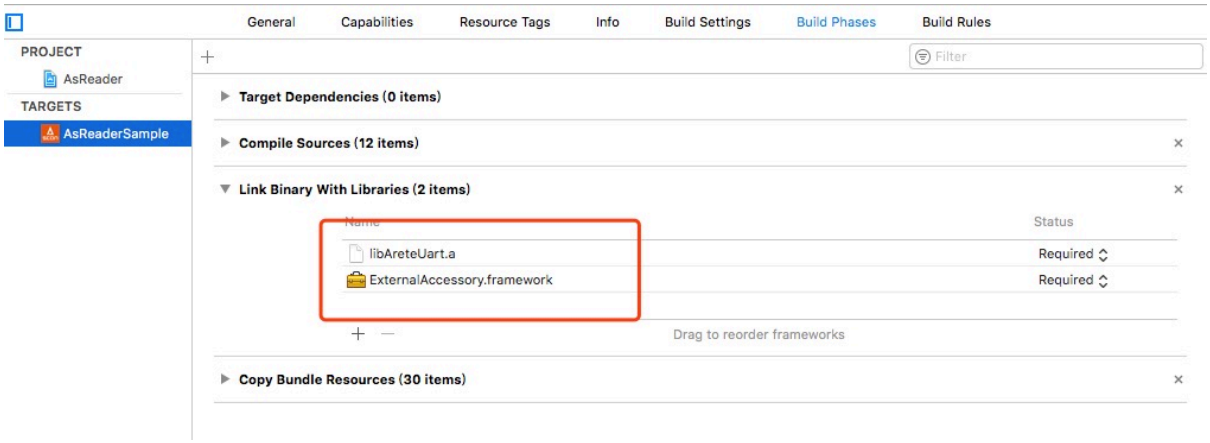
1.3.1 libAreteUart.a を追加します。



1.3.2「Add Other」をクリックし、libAreteUart.a を追加します。



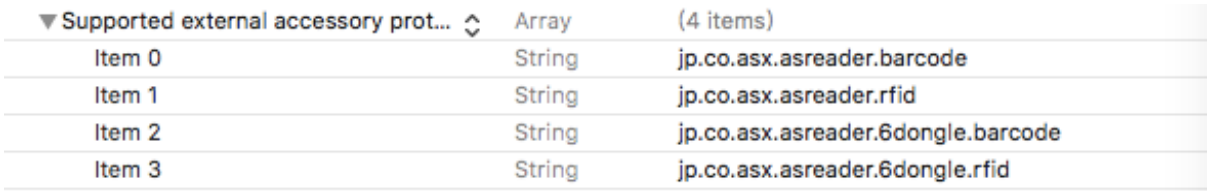
1.3.3 libAreteUart.a が追加されていることを確認します。



1.4 AsReader protocol の追加

info.plist に Supported external accessory protocols を追加し、以下を追加します。

- 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-030D, ASR-031D : jp.co.asx.asreader.6dongle.rfid



1.5 SDK のインポート

使用したいプロジェクトの*.m と*.h に、SDK を import します。

#import “RcpBarcodeApi.h”

1.6 注意事項

本 SDK を使用する時は，C++言語を使用している為、プロジェクトに.mm ファイルを追加するか、Xcode の設定に libc++を追加する必要があります。

（.mm ファイルの中身は空でよい）

2 BarcodeApi Class

サポートされている AsReader：ASX-510R、ASX-520R、ASR-010D、ASR-020D

2.1 init

```
- (id)init;
```

説明：RcpBarcodeApi を初期化します。
返り値：成功：RcpBarcodeApi クラスインスタンス
　　失敗：nil

2.2 open

```
- (BOOL)open;
```

説明：AsReader を起動します。
返り値：成功：YES
　　失敗：NO

2.3 isOpened

```
- (BOOL)isOpened;
```

説明：AsReader の起動状態を確認します。
返り値：起動：YES
　　未起動：NO

2.4 close

```
- (void)close;
```

説明：AsReader とのセッションを閉じます。

2.5 startReadBarcodes

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

説明：バーコードの読取りを開始します。
引数：mtnu:0x00 mtime:0x00 repeatCycle:0x00
返り値：成功：YES
　　失敗：NO

2.6 setReaderPower

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

説明： バーコードエンジンの電源を操作します。
引数： 電源 On ： YES / 電源 Off ： NO。
返り値：成功：YES
失敗：NO

2.7 setBeep

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

説明： AsReader のビープ音、バイブ、イルミネーションを設定します。
引数： beepOn ： On (0x01) / Off (0x00)
vibrationOn ： On (0x01) / Off (0x00)
illuminationOn ： On (0x01) / Off (0x00)

2.8 getSDKVersion

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

説明：SDK のバージョン情報を取得します。
返り値：バージョン（例：1.3.3）

3 BarcodeDelegate Class

サポートされている AsReader：ASX-510R、ASX-520R、ASR-010D、ASR-020D

3.1 readerConnected

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

説明 :setReaderPower または setReaderPowerOnWithBeep の後に Delegate から結果をコールバックします。
引数：電源 On 成功：0xFF / 電源 On 失敗：0x00
注意：本メソッドが Delegate からコールバックされるまで、その他 SDK メソッドを使用しないでください。

3.2 pluggedBarcode

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

説明：AsReader の接続状態が変化時、Delegate から接続状態をコールバックします。
引数：接続: YES
　　切断: NO

3.3 barcodeStringReceived

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

説明：AsReader が読み取った結果を Delegate からコールバックします。
呼び出し方法 :startReadBarcodes メソッドまたは AsReader 本体のトリガーボタン
引数：barcode：バーコード情報（NSString 型）。

3.4 batteryChargeReceived

```
- (void)batteryChargeReceived:(int)battery;
```

説明：AsReader 接続成功後、電池残量情報をコールバックします。（10s ごと一回）
引数： battery：0、25、50、75、100（%で表示する。例：現在の電池残量が 75 の場合、75%を表示する。）
実際の電池残量と各%：25%: 1～25%
　　50%: 26～50%
　　75%: 51～75%
　　100%: 76～100%

4 RfidApi Class

サポートされている AsReader：ASX-300R、ASR-030D、ASR-031D

4.1 getSDKVersion

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

説明：SDK のバージョン情報を取得します。
返り値：バージョン（文字列）例 1.0.1

4.2 init

```
- (id)init;
```

説明：RcpRfidApi を初期化します。
返り値：成功：RcpRfidApi クラスインスタンス
　　失敗：nil

4.3 open

```
- (BOOL)open;
```

説明：AsReader を起動します。
返り値：成功：YES
　　失敗：NO

4.4 isOpened

```
- (BOOL)isOpened;
```

説明：AsReader の起動状態を確認します。
返り値：起動：YES
　　未起動：NO

4.5 close

```
- (void)close;
```

説明：AsReader とのセッションを閉じます。

4.6 setReaderPower

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

説明：RFID モジュールの電源を操作します。
引数：電源 On ： YES　/　電源 Off ： NO。
返り値：成功：YES
　　失敗：NO

4.7 setBeep

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

説明： AsReader のビープ音、バイブ、イルミネーションを設定します。
引数： beepOn ： On (0x01) / Off (0x00)
　　vibrationOn ： On (0x01) / Off (0x00)
　　illuminationOn ： On (0x01) / Off (0x00)　※未使用

4.8 startReadTags

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

説明：RFID の読取りを開始します。
引数： mtnu：読み取りするタグの最大数
　　mtime：読取りの最大時間
　　repeatCycle：読取りの繰り返し回数
返り値：成功：YES
　　失敗：NO

4.9 startReadTagsWithRssi

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

説明：RFID の読取りを開始します。同時に RSSI データも読取ります。
引数：mtnu：読み取りするタグの最大数
　　mtime：読取りの最大時間
　　repeatCycle：読取りの繰り返し回数
返り値：成功：YES
　　失敗：NO

4.10 stopReadTags

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

説明：タグの読取りを停止します。
返り値：成功：YES
　　失敗：NO

4.11 getChannel

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

説明：RF チャンネルを取得します。
返り値：成功：YES
　　失敗：NO

4.12 setChannel

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

説明：AsReader に「Set current RF channel」コマンドを送信します。
注意：LBT の場合に使用してください。
引数：Channel(8 bit), Channel Offset for Miller Subcarrier (8 bit)
返り値：成功：YES
　　失敗：NO

4.13 getFhLbtParam

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

説明：FH と LBT の引数を取得します。
返り値：成功：YES
　　失敗：NO

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;
```

説明：FH と LBT の引数を設定します。
引数：readTime:読取り時間(ms)。
　　idleTime:アイドル時間(ms)。
　　carrierSenseTime: キャリア検知時間(ms)。
　　rfLevel:rf レベル (-dBm X 10)。
　　frequencyHopping:起用：0x01 以上/禁止：0x00。
　　listenBeforeTalk：起用：0x01 以上/禁止：0x00。
　　continuousWave:起用：0x01/禁止:0x00。
返り値：成功：YES
　　失敗：NO

4.15 getOutputPowerLevel

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

説明 :現在の Tx パワーレベル及び最大と最小 Tx パワーレベルを取得します。(取得された Tx パワーレベルは txPowerLevelReceived デリゲートを通して CommonReaderInfo クラスに値をセットします)
返り値：成功：YES
　　失敗：NO

4.16 setOutputPowerLevel

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

説明： Tx パワーレベルを設定します。
引数： power：Tx パワーレベル(日本版の Tx パワーレベル範囲：18～24dBm、日本版以外の Tx パワーレベル範囲：18～25dBm)。
返り値：成功：YES
失敗：NO

4.17 readFromTagMemory

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

説明：指定されるメモリの Type C タグデータを読取ります。
引数：accessPassword：アクセスパスワード
epc：タグ
memoryBank：RFU (0x00)、 EPC (0x01)、 TID (0x02)、 User (0x03)
startAddress：スタートアドレス
dataLength: データの長さ
返り値：成功：YES
失敗：NO

4.18 getSession

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

説明：session を取得します。
返り値：成功：YES
失敗：NO

4.19 setSession

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

説明：session を設定します。
引数：session:S0(0x00), S1(0x01), S2(0x02), S3(0x03)
返り値：成功：YES
失敗：NO

4.20 getAnticollision

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

説明：アンチコリジョンモードを取得します。
返り値：成功：YES
　　失敗：NO

4.21 setAnticollision

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

説明：アンチコリジョンモードを設定します。
引数：アンチコリジョンモード：固定 Q(0x00)，ダイナミック Q(0x01)
　　qStart:スタート Q
　　qMax:最大 Q
　　qMin:最小 Q
返り値：成功：YES
　　失敗：NO

4.22 writeToTagMemory

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

説明：タグを書き込みます。
引数：epc：タグの EPC。
　　memoryBank：RFU (0x00)，EPC (0x01)，TID (0x02)，User (0x03)
　　startAddress：スタートアドレス
　　dataToWrite：書き込みデータ
返り値：成功：YES
　　失敗：NO

4.23 killTag

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

説明：タグをキルします。
注意：タグをキルする時、必ずパスワードを設定してください。
引数：password：パスワード，　0x00000000 に設定された場合はタグキルは無効。
　　epc：タグの EPC。
返り値：成功：YES
　　失敗：NO

4.24 lockTagMemory

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

説明：タグをロックします。
注意：タグをロックする時、必ずパスワードを設定してください。
引数：accessPassword：アクセスパスワード。0x00000000 に設定された場合はタグロック無効。
　　epc：タグの EPC
　　lockData：ロックデータ
返り値：成功：YES
　　失敗：NO

4.25 setStopConditionMtnu

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

説明：読み取りの停止条件を設定します。
引数：　mtnu：読み取りするタグの最大数
　　mtime：読取りの最大時間
　　repeatCycle：読取りの繰り返し回数
返り値：成功：YES
　　失敗：NO

4.26 setOptimumFrequencyHoppingTable

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

説明：最適な FH チャンネルテーブルを自動設定します。
返り値：成功：YES
　　失敗：NO

4.27 SetFrequencyHoppingMode

```
- (BOOL)SetFrequencyHoppingMode:(uint8_t)mode;
```

説明：FH モードを設定します。
引数：FH モード：Normal Mode(0x00), Smart Hopping Mode(0x01)
返り値：成功：YES
　　失敗：NO

4.28 updateRegistry

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

説明：レジストリを更新します。
返り値：成功：YES
　　失敗：NO

4.29 writeToTagMemory

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

説明：タグデータを書き込みます。
引数：epc：タグの epc
　　dataToWrite：書込みデータ
返り値：成功：YES
　　失敗：NO

4.30 startReadTagsRFM

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

説明：RFID 温度タグ/湿度タグの読み取りを開始します。
引数：codeType：タグタイプ：温度タグ：0x03/湿度タグ：0x02
　　mtnu: 読み取りするタグの最大数
　　mtime: 読取りの最大時間、単位：s
　　repeatCycle: 読取りの繰り返し回数
返り値：YES：メソッド実行に成功
　　NO：メソッド実行に失敗

5 RfidDelegate Class

サポートされている AsReader：ASX-300R、ASR-030D、ASR-031D

5.1 pluggedRfid

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

説明：AsReader 接続状態変化時、実行結果をコールバックします。
引数：成功：YES
　　失敗：NO

5.2 pcEpcReceived

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

説明：AsReader のトリガーを押す、それとも「startReadTags」実行に伴い、スキャンされたデータをコールバックします。
引数：pcEpc:タグデータ

5.3 pcEpcRssiReceived

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

説明：「startReadTagsWithRssi」の実行結果をコールバックします。
引数：pcEpc：タグデータ
　　rssi：rssi データ

5.4 readerConnected

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

説明：「setReaderPower」の実行結果をコールバックします。
引数：status：状態コード；成功(0x00)，失敗(0x00 以外)

5.5 readerConnected

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

説明：電源をオンにし、「setpower on」の実行結果をコールバックします。

5.6 errReceived

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

説明： コマンド実行のエラー情報を取得します。コマンド実行エラーが発生時、コールバックします。（本メソッドはBarcode及びRFID共通）。
引数：errcode：エラーコード
0x09 ： タグメモリの読取り失敗
0x10 ： データ書込み失敗
0x0B ：「Read Type C Tag ID Multiple」有効
0x0D ：「Read Type C Tag ID Multiple」モジュールではない
0x0E ： 引数無効
0x12 ： タグキル失敗
0x13 ： タグロック失敗
0x15 ： タグの読取り失敗
0x18 ： サポートしないコマンド
0xFF ： CRCエラー

5.7 errDetailReceived

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

説明:詳細なエラーメッセージを取得します。コマンド実行エラーが発生する時、本メソッドをコールバックします。（本メソッドはバーコード及びRFID共通）
引数：errCode：エラーコード

- Error code (8-bit)
  - 0x01 ： リーダー電源のコントロール失敗
  - 0x02 ： リーダーのコントロール失敗
  - 0x03 ： リーダーの情報を取得失敗
  - 0x07 ： スキャン区域を取得失敗
  - 0x08 ： スキャン区域を設定失敗
  - 0x09 ： タグメモリの読取り失敗
  - 0x0A ： 自動読取りの操作失敗
  - 0x0B ： 自動読取りの操作
  - 0x0C ： 自動読取りが止められない
  - 0x0D ： 自動読取りモードではない
  - 0x0E ： 引数無効
  - 0x10 ： データ書込み失敗
  - 0x11 ： データクリア失敗
  - 0x12 ： タグキル失敗
  - 0x13 ： タグロック失敗
  - 0x15 ： タグを検出されなかった
  - 0x17 ： サポートしないコマンド
  - 0x18 ： 未定義のコマンド
  - 0x19 ： リーダーリセット失敗

- 0xFF ： CRC エラー

➢ Sub Error code (8-bit)

- 0x01 ： サポートしない
- 0x02 ： 権限不足
- 0x03 ： メモリオーバーラン
- 0x04 ： メモリロックされた
- 0x05 ： 暗号スイートエラー
- 0x06 ： コマンドがカプセル化されなかった
- 0x07 ： レスポンスバッファオーバーフロー
- 0x08 ： セキュリティタイムアウト
- 0x0B ： 電力不足
- 0x0F ： 指定されていないエラー
- 0x11 ： センサースケジュール設定
- 0x12 ： タグ応答なし
- 0x13 ： 測定タイプサポートしない
- 0x80 ： タグを検出されなかった
- 0x81 ： 処理の取得失敗
- 0x82 ： パスワードアクセス失敗
- 0x90 ： CRC エラー
- 0x91 ： レジストリのクリア失敗
- 0xA1 ： SPI 失敗
- 0xA2 ： Rx タイムアウト
- 0xB1 ： レジストリの更新失敗
- 0xB2 ： レジストリの書込み失敗
- 0xA0 ： レジストリが存在しない
- 0xA3 ： UART 失敗
- 0xB0 ： I2C 失敗
- 0xB3 ： GPIO 失敗
- 0xE0 ： サポートしないコマンド
- 0xE1 ： 未定義のコマンド

- 0xE2 ： 引数無効
- 0xE3 ： パラメータが高すぎる
- 0xE4 ： パラメータが低すぎる
- 0xE5 ： 自動読取り失敗
- 0xE6 ： 自動読取りモードではない
- 0xE7 ： 最後のレスポンスを読取り失敗
- 0xE8 ： リーダーリセット失敗
- 0xE9 ： RFID ロックのコントロール失敗
- 0xEA ： タグメモリの読取り失敗

5.8 frequencyHoppingModeReceived

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

説明：「getFreqHoppingTable」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)，失敗(0x00 以外)

5.9 regionReceived

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

説明：「getRegion」の実行結果をコールバックします。
引数： region：地域

5.10 channelReceived

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

説明： 「getChannel」の実行結果をコールバックします。
引数： channel：チャンネル
channelOffset：チャンネルオフセット

5.11 fhLbtReceived

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

説明：「getFhLbtParam」の実行結果をコールバックします。
引数： fhLB：FH と LBT

5.12 tagMemoryReceived

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

説明：「writeToTagMemory」の実行結果をコールバックします。
引数： data：タグデータ

5.13 anticolParamReceived

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

説明：「getAnticollision」の実行結果をコールバックします。
引数： mode：固定 Q：0x00/ダイナミック Q：0x01。
start: スタート Q
max:最大 Q
min:最小 Q

5.14 batteryChargeReceived

```
- (void)batteryChargeReceived:(int)battery;
```

説明：AsReader 接続成功後、電池残量情報をコールバックします。（10s ごと一回）
引数： battery：0、25、50、75、100（%で表示する。例：現在の電池残量が 75 の場合、75%を表示する。）
実際の電池残量と各%：25%: 1～25%
50%: 26～50%
75%: 51～75%
100%: 76～100%

5.15 startedReadTags

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

説明：「startReadTags」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)，失敗(0x00 以外)

5.16 didSetOutputPowerLevel

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

説明：「setOutputPowerLevel」の実行結果をコールバックします。
引数： status：状態コード；成功(0x00)，失敗(0x00 以外)

5.17 writedReceived

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

説明：「writeTagMemory」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)，失敗(0x00 以外)

5.18 stoppedReadTags

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

説明：「stopReadTags」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)，失敗(0x00 以外)

5.19 lockedReceived

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

説明：「lockTagMemory」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)，失敗(0x00 以外)

5.20 didSetFhLbtReceived

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

説明： 「setFreqHoppingTable」の実行結果をコールバックします。
引数： statusCide：成功(0x00)，失敗(0x00 以外)

5.21 didSetAntiColModeReceived

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

説明：「setAnticollision」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)，失敗(0x00 以外)

5.22 sessionReceived

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

説明：「getSession 」の実行結果をコールバックします。
引数： session：S0(0x00)，S1(0x01)，S2(0x02)，S3(0x03)

5.23 didSetStopConditionMtnu

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

説明：「setStopCondition」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)，失敗(0x00 以外)

5.24 didSetOptiFreqHPTable

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

説明： 「setFreqHoppingTable」の実行結果をコールバックします。
引数： statusCode：設定開始(0x00),設定完了(0x01)

5.25 didSetFreqHPMode

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

説明：「setFreqHoppingTable」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)， 失敗(0x00 以外)

5.26 didSetSession

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

説明：「setSession」の実行結果をコールバックします。
引数： statusCode：状態コード；成功(0x00)， 失敗(0x00 以外)

5.27 txPowerLevelReceived

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

説明： 「getOutputPowerLevel」の実行結果をコールバックします。
コールバック後 RFID TX Power 値を commonReadInfo オブジェクトに保存します。
fRFIDpower： 現在のアウトプットパワー
fRFIDpowerMax： 設定できる最大のアウトプットパワー
fRFIDpowerMin： 設定できる最小のアウトプットパワー

5.28 pcEpcSensorDataReceived

注意：温度タグ/湿度タグを読み取りのデリゲートメソッド

```
- (void)pcEpcSensorDataReceived:(NSData *)pcEpc sensorData:(NSData *)sensorData;
```

説明: 呼び出しメソッド:startReadTagsRFM の実行結果をコールバックします。
引数： pcEpc：温度タグ/湿度タグデータ

sensorData：温度/湿度データ

サンプルコード：

```
- (void)pcEpcSensorDataReceived:(NSData *)pcEpc sensorData:(NSData
*)sensorData
{
    int codeType;//タグタイプ，2（湿度タグ）/3（温度タグ）
    int onChipRssiCodeValue;// タグチップRSSIデータ
    int sensorCodeValue;// 温度/湿度データ（16進数）
    double calcTemp;// 温度（摂氏）
    NSMutableString *tmptagid;// タグpcepcデータ（16進数）

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

    //pcepc NSData変換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 ];

    //温度・湿度データ解析
    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;
    }
}
```