Neo Pen iOS SDK

with sample application

< Tutorial Document >

Document Version: v1.5.1

Date: 2016.2.16

NeoLAB Convergence Inc.

Table of Contents

Revision history

17.DEC.2014	NISDK v1.0 and document release	L. Park/K.You
3.AUG.2015	NISDK v2.0.1 and document v1.1.2 release	L.Park
12.NOV.2015	NISDK v2.1 and document v1.2 release Add PUI (paper ui) for email Add delegate method for setting transferable note IDs Add delegate method for offline data note list count	L.Park
26.NOV.2015	NISDK v2.2 and document v1.3 release Add NJPenCommManagerNewPeripheral protocol and relative delegate method for discovered peripherals list during BT scan Add delegate method for notifying if a pen has been connected by other apps	L.Park
8.DEC.2015	NISDK v2.3 and document v1.4 release Add delegate method for getting a path to offline raw files before parsed Add command for setting transferable note IDs according to section and owner IDs from property list Add pen status none to BT scan Add “Change canvas color” to n2sample menu Add dot checker for offline data to “parseOfflineDots” delegate method in n2sample	L.Park
21.DEC.2015	NISDK v2.4 and document v1.5 release Replace delegate method for offline data note list count Add method for reading battery level and memory space of a pen Add index and additional description for header files	L.Park

1. Overview

This application note is for Neo Pen iOS SDK (NISDK framework ver2.1).
Developing Neo Pen application requires three entities. They are a Neo Pen, a notebook with nCode and iPhone.

1.1 Neo Pen

While users write on notebooks with Neo Pen as they do with normal pens, Neo Pen reads coordinates on notebook and send them to iPhone. To send coordinates, Neo Pen and iPhone are connected to each other via Bluetooth. The firmware version of Neo Pen should be above v1.03.137 or above for the NISDK framework ver2.1.

1.2 Notebook with nCode

Neo Pen reads coordinates of strokes which a user is writing on a notebook. For Neo Pen to read coordinates the notebook has nCodes printed on each pages. By reading the codes Neo Pen retrieves some information. It involves,

- Coordinates of each stroke
- Note Type : Unique ID given to a specific notebook type.
- Page Number : Page number which the stroke is written on

1.3 iPhone

Neo Pen can communicate with any devices equipped Bluetooth. This application note uses iPhone for its counterpart. More specifically they are connected via Bluetooth Low Energy protocol(BTLE). iPhone's role in BTLE communication is "central" device and Neo Pen's is "peripheral". Usual application may requires iPhone to be responsible for,

- Connecting/Disconnecting to Neo Pen via BTLE
- Saving strokes in its storage
- Rendering the stroke

1.4 URL scheme

The URL of the app is “neonates” for URL scheme. You can call the app from the app version 1.6.

2. SDK structure

The iOS SDK sources are grouped under n2base folder. There are 7 sub-folders under the folder.

Name	Description
penComm	Bluetooth communication and protocol parser
pageData	Core data models representing writing data
models	Data models
manager	Classes handling data models
common	Strings, definitions
utils	Utility functions
background	pdf files to be used as notebook background images

2.1 penComm

Applications use NJPenCommManager to send data to Neo Pen and control BTLE connection. It is designed to be used as singleton across the application. The best practice to get instance of the Class looks like below.

```
pencommManager = [NJPenCommManager sharedInstance];
```

NJPenCommManger handles BTLE protocol related communications such as scan, connecting, disconnecting, services, characteristics, etc.
NJPenCommParser handles protocol defined for communication between Neo Pen and iPhone.

2.2 pageData

The activity writing on a notebook is delivered to an iOS device as a group of coordinate representing a dot. Each dot has x, y coordinate on the page of the notebook. Each group may have one or more dots. And we call the group of dots as stroke.
A stroke is defined as a set of dots sampled by Neo Pen from the moment the pen touches the notebook page(Pen Down) to the moment the pen is pulled up to be separated from the paper(Pen Up). NJStroke represents a stroke.
While a user keeps writing on a page, there should be a lot of strokes delivered to an iOS device. NJPage is container of strokes written on the same page.

3. Getting Started with N2 iOS SDK

This chapter covers what you need to go through in order to be able to start using the N2 SDK for iOS.

The 4 easy steps are as follows:

- Step 1: Prerequisites
- Step 2: Prepare the N2 SDK for iOS
- Step 3: Add the N2 SDK to your Xcode project
- Step 4: Start coding

3.1 Step 1 : Prerequisites

- OS X is required for all iOS development
- You need Xcode. If you don't have it, you can get it from the App Store.

Note: The N2 SDK v2.1 for iOS supports iOS 8.x and higher.

3.2 Step 2: Prepare the N2 SDK for iOS

The N2 SDK for iOS is a framework which is called as "NISDK.framework".
You will be provided with NISDK.framework separately or you can get it from the path of the sample code, "n2sample/NISDK.framework"

3.3 Step 3: Add the N2 SDK to your Xcode Project

- Add the N2 SDK for iOS to your Xcode project
    - Create a new project in Xcode
    - Select the NISDK.framework and drag it into the "Frameworks" group in Xcode
- Add a framework (libz.dylib)

3.4 Step 4: Start coding

- Add the N2 SDK for iOS header file
    - #import <NISDK/NISDK.h>

4. Using N2 iOS SDK

4.1 Controlling BTLE connection(connect/disconnect)

4.1.1 Connect/Disconnect

With an instance of NJPenCommManager,

- [pencommManager btStart]
  Scan Neo Pen and connect to the pen found.
- [pencommManager disConnect]
  Disconnect from Neo Pen
- [pencommManager btStop]
  Stop scanning Neo Pen.
- [pencommManager hasPenRegistered]
  a flag for whether or not pen was registered. If a pen is registered, it will be YES. If not, it will be NO.
- [pencommManager isPenConnected]
  a flag for whether or not pen was connected. If a pen is connected, it will be YES. If not, it will be NO.
- [pencommManager resetPenRegistration]
  Reset pen registration. It will disconnect with Neo Pen and delete a uuid registered. Also, set [pencommManager hasPenRegistered] to NO.
- [pencommManager regUuid]
  Return a uuid string for a pen registered and connected
- NJPenCommManPenConnectionStatus penConnectStatus
  -. Show a pen connection status.
  -. typedef NS_ENUM (NSInteger, NJPenCommManPenConnectionStatus) {
  NJPenCommManPenConnectionStatusNone,
  NJPenCommManPenConnectionStatusScanStarted,
  NJPenCommManPenConnectionStatusConnected,
  NJPenCommManPenConnectionStatusDisconnected
  };
  -. The pen connection status will be notified with "NJPenCommManagerPenConnectionStatusChangeNotification" and it can be read with a status data as follows.
  NSInteger penConnctionStatus = [[[notification userInfo] valueForKey:@"info"] integerValue];

4.1.2 Input password

The default pen password is "0000". But, if a pen doesn't have the same password to the app, a correct password via a customized keypad should be input. After comparing a pen password, the pen connection process will be completed. Please refer to the 4.4.1 for a relevant method.

4.2 Stroke Data

If the application wants to get notified of the stroke in real time, the application requires to register NJPenCommParserStrokeHandler to NJPenCommManager by calling setPenCommParserStrokeHandler.

If there is no stroke handler registered when a stroke is delivered from Neo Pen, penCommManager will notify with NJPenCommParserPageChangedNotification. This is to give an application a chance to register NJPenCommParserStrokeHandler to handle subsequent stroke data in real time. Usual application of handling real time stroke data is rendering the strokes on a view.

- (void) processStroke:(NSDictionary *)stroke
  -. It is called after a dictionary is set for stroke data and pen up/down status. A key 'type' is used to distinguish stroke and updown. A key 'node' is used for a node information (x, y coordinate). And, a key 'status' has information about 'up' or 'down' for pen. The sample application uses it to render stroke.
- (void)notifyPageChanging
  This is to let the application informed there is a data for a new page. The application use this information to stop rendering subsequent stroke until getting notified with page opened notification.
- NJPage
  Class which includes the page information such as stroke data(NJStroke), paper size, notebook id, page number and so on. A NJPage object should be created before a page canvas is established. Please refer to NJPage.h for more details.
- NJStroke
  Class which includes the node information such as x, y coordinates.
  Please refer to NJStroke.h for more details.
- NJNode
  Class for the node information such as x, y coordinates, pressure and time.
  Please refer to NJNode.h for more details.
- activeNoteBookId, activePageNumber
  -. activeNoteBookId is needed to know the ID of the note book which is being used and activePageNumber is needed to know the number of the page on which is being used
  -. activeNoteBookId and activePageNumber are necessary for a page canvas to be created and stroke data can be added on the page canvas while they are being written.
  -. They can be read by the below delegate method of the protocol "NJPenCommParserStrokeHandler".
  -. (void) activeNoteId:(int)noteId pageNum:(int)pageNumber sectionId:(int)section ownderId:(int)owner pageCreated:(NJPage *)page;
  : (int)noteId, (int)pageNumber, (int)section and (int)owner are note id, page page number, section id and owner id for the active page. And, (NJPage *)page is a page object which was created for the active page.
  But, for the first stroke as soon as a pen is connected, they can be read by the below delegate method of the protocol, "NJPenCommParserStartDelegate".
  -.(void) activeNoteId:(int)noteId pageNum:(int)pageNumber : note id and page number for the page where the first stroke was added.
  -.(void) firstStrokePage: (NJPage *)fPage : a page object for the page where the first stroke was added.
  -.(void) setPenCommNoteIDList :
  [pencommManager setAllNoteIdList] if you want to get all notes to be written on notes.
  [pencommManager setNoteIdListFromPList] if you want have notes list from "note_support_list.plist" applied and written on the note from the list.
  [pencommManager setNoteIdListSectionOwnerFromPList] if you want

have all notes under section and owner ids from “note_support_list.plist” applied and written on the note. It means even though some note ids are not in the property list, but their section ids and owner ids are in the list, they can be written.

- (UInt32)setPenColor
  This is a delegate method under NJPenCommParserStrokeHandler protocol.
  Strokes color on canvas can be set through this method. Please refer to a new menu “Change canvas color” in the n2sample code for more details.

4.3 Rendering example

4.3.1 Simple rendering

The sample application draw lines to draw strokes in real time. It is implemented in NJPageCanvasController.m. Whenever processStroke is called it draw line with the dot received.

- (UIImage *) drawStroke: (NJStroke *)stroke withImage:(UIImage *)image
  size:(CGRect)bounds scale:(float)scale offsetX:(float)offset_x
  offsetY:(float)offset_y;

4.3.2 Fountain Pen style rendering

If the application receives pen up, it means there is a complete stroke received. With the stroke data the sample application renders a stroke emulating fountain pen.
The sample application catches pen up state in processStroke and update a view with this rendering.

4.4 Pen password

4.4.1 Setting NJPenPasswordDelegate and implementing a callback method

- Setting the delegate

[[NJPenCommManager sharedInstance] setPenPasswordDelegate:self];

- Callback method

- (void) penPasswordRequest:(PenPasswordRequestStruct *)request;
  : retry count and reset count which were transmitted can be read from request.

4.4.2 Pen password input

[[NJPenCommManager sharedInstance] setBTComparePassword:password];

4.4.3 Pen password change

[[NJPenCommManager sharedInstance] changePasswordFrom:self.savedPin To:self.currentPin];

4.4.4 Notification

If a password input is correct, the notification NJPenCommManagerPenConnectionStatusChangeNotification will be sent.

And if a password change is succeeded, the notification NJPenCommParserPenPasswordSutupSuccess will be sent.

4.5 Setting

4.5.1 Auto power on and sound

[[NJPenCommManager sharedInstance] setPenStateAutoPower:pAutoPwer Sound:pSound];

4.5.2 Shutdown Timer

[[NJPenCommManager sharedInstance] setPenStateWithAutoPwrOffTime:autoPwrOff];

4.5.3 Pen sensor pressure tuning

[[NJPenCommManager sharedInstance] setPenStateWithPenPressure:penPressure];

4.6 Offline Sync

4.6.1 Setting NJOffineDataDelegate and requesting Offline Sync file list

- The method by which requestOfflineFileList is performed.

[[NJPenCommManager sharedInstance] setOfflineDataDelegate:self];

- callback method for offline data file list. Note list can be acquired from noteListDic.

(void) offlineDataDidReceiveNoteList:(NSDictionary *)noteListDic

- callback method for offline data file note list count of a section and an owner ID

(void) offlineDataDidReceiveNoteListCount:(int)noteCount
ForSectionOwnerId:(UInt32)sectionOwnerId

4.6.2 Offline Sync file data request and receiving stroke data from pen

- requesting offline sync file data

(BOOL) requestOfflineDataWithOwnerId:(UInt32)ownerId noteId:(UInt32)noteId;

- callback method for offline sync data status while the data is being transmitted

(void) offlineDataReceiveStatus:(OFFLINE_DATA_STATUS)status percent:(float)percent

- receiving stroke data from pen

(void) parseOfflineDots:(NSData *)penData startAt:(int)position
withFileHeader:(OffLineDataFileHeaderStruct *)pFileHeader
andStrokeHeader:(OffLineDataStrokeHeaderStruct *)pStrokeHeader
: please refer to NeoPenService.h for the structures for "OffLineDataFileHeaderStruct" and "OffLineDataStrokeHeaderStruct".
In addition, you can get section ID from nOwnerId which consists of section ID and

Owner ID. For example, if nOwnerId is 50331675 and is converted to hex from decimal, it will be 0x300001B. The MSB 8bits is section ID (0x3) and the LSB 24bits is owner ID (0x1B, decimal:27).

- receiving a path to offline raw file before parsed.
  (void) offlineDataPathBeforeParsed:(NSString *)path
  : The offline raw file is the same as a pen has. If you want to save it in your app locally, you can use this method. However, it should be saved as soon as it is transmitted from the pen, because an offline file gets to be replaced by a next one in the same path.

4.7 Upgrade pen firmware

4.7.1 Setting NJFWUpdateDelegate

[[NJPenCommManager sharedInstance] setFWUpdateDelegate:self];

4.7.2 Sending pen firmware file to pen

[[NJPenCommManager sharedInstance] sendUpdateFileInfoAtUrlToPen:filePath];

4.7.2 Suspending sending pen firmware file

- canceFWUpdate flag should be set into YES if stopping sending pen firmware needs to be done.

[NJPenCommManager sharedInstance].cancelFWUpdate = YES;

4.7.3 Read a firmware version

- [[NJPenCommManager sharedInstance] getFWVersion]
  Return a Firmware version string for a pen connected

4.7.4 Firmware update procedure

- Please refer to “NJFWUpdateViewController.m” of n2sample application for how firmware update is implemented.

1. delegate setting as follows when a viewcontroller for firmware update starts.
   =>[[NJPenCommManager sharedInstance] setFWUpdateDelegate:self];
2. read a version number and a location string from the json file of the following path
   http://one.neolab.kr/resource/fw/f1xx_firmware.json
3. download a firmware version from the following server path.
   http://one.neolab.kr/resource/fw/ + location (from 2)
4. send the firmware version file downloaded from 2 to a N2 pen via the following API.
   =>[[NJPenCommManager sharedInstance] sendUpdateFileInfoAtUrlToPen:filePath]
5. you can get to know how much the firmware file transmits to the pen from the app via the following method.
   =>(void)fwUpdateDataReceiveStatus:(FW_UPDATE_DATA_STATUS)status percent:(float)percent
6. The blue led of the pen blinks (it means its firmware is being updated) when the pen resets by pressing a power button, if the firmware file transmits 100% successfully.

4.8 Pen status

4.8.1 Setting NJPenStatusDelegate and sending setPenState

- setting the delegate

[[NJPenCommManager sharedInstance] setPenStatusDelegate:self];

- sending pen state

[[NJPenCommManager sharedInstance] setPenState];

4.8.2 callback method

(void) penStatusData:(PenStateStruct *)penStatus

4.9 Transferable note Id

4.9.1 Setting transferable note id

[[NJPenCommManager sharedInstance] setNoteIdListFromPList];

4.10 Pen tip led color

4.10.1 Setting pen tip color led

[[NJPenCommManager sharedInstance] setPenStateWithRGB:penColor];

4.11 BT list for discovered peripherals

It returns peripherals array and uuid array discovered during BT peripherals scan.

4.11.1 Setting NJPenCommManagerNewPeripheral

- setting the delegate
  [[NJPenCommManager sharedInstance] setHandleNewPeripheral:self];
- BT scan command to have peripherals array and uuid array returned. Timer should be set after performing this command
  [[NJPenCommManager sharedInstance] btStartForPeripheralsList];
- The arrays should be read after timer expiry. Discovered peripherals will be collected during the time set by timer. You can check it with sample application menu “BT List”.
  [[NJPenCommManager sharedInstance] discoveredPeripherals];
- Command for connection with a pen which is selected
  [[NJPenCommManager sharedInstance] connectPeripheralAt: (NSInteger)index];

4.11.2 callback method

(void) connectionResult:(BOOL)success

: It returns connection result for “connectPeripheralAt” method.

4.12 Paper UI for email

An email client view will be presented, if you mark on an email icon of a note.

4.12.1 Setting NJPenCommParserCommandHandler

- setting the delegate

[[NJPenCommManager sharedInstance] setPenCommParserCommandHandler:self];

4.12.2 callback method

(void) sendEmailWithPdf

This delegate method will be called if an email icon is marked on a note.

4.12.2 Generating 'note_pui_info.plist'

Please refer to the chapter 5 for how to set up the plist file.

4.13 Notification when a pen has already been connected by other apps

App will be notified with a delegate callback method, if a pen has already been connected by the one of other apps installed.

4.13.1 Setting NJPenCommParserCommandHandler

- setting the delegate

[[NJPenCommManager sharedInstance] setPenCommParserCommandHandler:self];

4.13.2 callback method

(void) penConnectedByOtherApp:(BOOL)penConnected

Boolean penConnected will be YES, if a pen has already been occupied by others.

If it is YES, your app will be able to notify with a pop-up message, for instance, "Your pen has been connected by the one of other apps. Please disconnect it from the app and please try again"

4.14 Battery level and used memory space of a pen

The current battery level and the used memory space of the pen can be read with the following method. Please refer to 'getPenBatteryLevelAndMemoryUsedSpace' or n2sample app menu 'Battery Level and Memory Space' at n2sample/NJViewController.m/ for more details.

(void) getPenBattLevelAndMemoryUsedSize:(void (^)(unsigned char remainedBattery, unsigned char usedMemory))completionBlock

5. Property list files creation

5.1 note_support_list.plist

This is used for setting transferable note IDs when a pen connection is initialized and established. A N2 pen can't be written on the notes which IDs are not included in the list. Please add note IDs which you want to use.

5.2 note_paper_info.plist

A new note can be added to the property list as the following steps.

1. Go to n2sample/Resources/note_support_list.plist
   Add a new note id to the end line of section 3, owner 27 if its section id is 3 and owner id is 27.

2. Go to n2sample/Resource/note_paper_info.plist

Name “sectionId_ownerId_0NoteId” and add it to the plist.
Add the number of pages, width, height, startX, startY, startPageNumber and bgFileName of the new note.

3. Add the new note background pdf file to n2sample/Resources/background directory.
Add the new note cover image to n2sample/Resources and should get the cover image for the new note id in your app.

5.3 note_pui_info.plist

It can be edited with the following steps.

1. Go to n2sample/Resource/note_pui_info.plist
2. Name “note ID” and add it to the plist.
3. Add page number, startX, startY, width, height of an email icon on a page.

6. Sample application

You can test the sample application as follows.

1. There is a menu button on the sample application. If it is pressed, you can see a menu list.
2. Select “register” menu to register the pen by pressing its power button for 3 seconds (you can see a blue led is blinking if the power button is pressed for 3 seconds). If it is registered and connected successfully, you can see a white led on from the pen. This menu will be changed into “connect” if the pen is registered.
3. The menu list should be performed after the pen is connected with the sample app.

Also, You can test offline sync (data transfer from a pen) with the sample application as follows.

1. Turn on a pen by pressing a power button and write on a note with the pen.
2. There is a menu button on the sample application. If it is pressed, you can see a menu list.
3. Select “connect” menu to connect the pen.
4. Select “OfflineData list” to see the note list where stroke data was added and they were saved in the pen at the above number 1 step.
5. Select “Show OfflineData” to see the page which was offline-synchronized into the sample application.

7. API index and description for header files

7.1 NJPenCommManager.h

```
@protocol NJPenCommManagerNewPeripheral
@optional
- (void) connectionResult:(BOOL)success;
@end
```

: Please refer to section 4.11 in page 12.

```
@protocol NJOfflineDataDelegate <NSObject>
- (void) offlineDataDidReceiveNoteList:(NSDictionary *)noteListDictionary;
- (void) parseOfflineDots:(NSData *)penData startAt:(int)position
withFileHeader:(OffLineDataFileHeaderStruct *)pFileHeader
       andStrokeHeader:(OffLineDataStrokeHeaderStruct *)pStrokeHeader;
@optional
- (void) offlineDataReceiveStatus:(OFFLINE_DATA_STATUS)status percent:(float)percent;
- (void) offlineDataDidReceiveNoteListCount:(int)noteCount
          ForSectionOwnerId:(UInt32)sectionOwnerId;

- (void) offlineDataPathBeforeParsed:(NSString *)path;
@end
```
: Please refer to section 4.6 in page 10.

```
@protocol NJPenCalibrationDelegate <NSObject>
@optional
- (void) calibrationResult:(BOOL)result;
@end
```
: It is not used

```
@protocol NJFWUpdateDelegate <NSObject>
@optional
- (void) fwUpdateDataReceiveStatus:(FW_UPDATE_DATA_STATUS)status percent:(float)percent;
@end
```
: Please refer to section 4.7 in page 11.

```
@protocol NJPenStatusDelegate <NSObject>
- (void) penStatusData:(PenStateStruct *)data;
@end
```
: Please refer to section 4.8 in page 11.

```
@protocol NJPenPasswordDelegate <NSObject>
- (void) penPasswordRequest:(PenPasswordRequestStruct *)data;
@end
```
: Please refer to section 4.4 in page 9.

```
@property (strong, nonatomic) NSMutableArray *discoveredPeripherals;
```
: Please refer to section 4.11 in page 12.

```
@property (nonatomic) BOOL cancelFWUpdate;
```
: Please refer to section 4.7 in page 11.

```
@property (nonatomic, readwrite) NJPenCommManPenConnectionStatus penConnectionStatus;
```
: Please refer to section 4.1 in page 7.

```
@property (nonatomic, readonly) BOOL isPenConnected;
```
: Please refer to section 4.1 in page 7.

@property (nonatomic, readwrite) BOOL hasPenRegistered;
: Please refer to section 4.1 in page 7.

@property (nonatomic, readonly) NSString *regUuid;
: Please refer to section 4.1 in page 7.

@property (nonatomic, readonly) NSString *penName;
: You can get this pen name which your pen has been transmitted as BT_ID, when it is registered.

@property (nonatomic) NSMutableArray *macArray;
: You can get mac list from pens around you during a specific time you set, when peripherals are scanned.

+ (NJPenCommManager *) sharedInstance;
: Please refer to section 2.1 in page 5.

- (void) setPenCommParserStrokeHandler:(id<NJPenCommParserStrokeHandler>)strokeHandler;

: Please refer to section 4.2 in page 7.

(void)setPenCommParserCommandHandler:(id<NJPenCommParserCommandHandler>)commandHandler;
: Please refer to section 4.12 in page 12.

- (void) setPenCommParserPasswordDelegate:(id<NJPenCommParserPasswordDelegate>)delegate;
: It is not used.

- (void) setPenCommParserStartDelegate:(id<NJPenCommParserStartDelegate>)delegate;
: Please refer to section 4.2 in page 8.

- (void) setOfflineDataDelegate:(id)offlineDataDelegate;
: Please refer to section 4.6 in page 10.

- (void) setPenCalibrationDelegate:(id)penCalibrationDelegate;
: It is not used.

- (void) setFWUpdateDelegate:(id)fwUpdateDelegate;
: Please refer to section 4.7 in page 11.

- (void) setPenStatusDelegate:(id)penStatusDelegate;
: Please refer to section 4.8 in page 11.

- (void) setPenPasswordDelegate:(id)penPasswordDelegate;
: Please refer to section 4.4 in page 9.

//Public API
- (void) btStart;
: Please refer to section 4.1 in page 7.

- (void) btStartForPeripheralsList;
: Please refer to section 4.11 in page 12.

- (void) btStop;
: Please refer to section 4.1 in page 7.

- (void) disConnect;
: Please refer to section 4.1 in page 7.

- (void) setPenState;
: Please refer to section 4.8 in page 12.

- (void)setNoteIdListFromPList;
: Please refer to section 4.2 in page 8.

- (void)setAllNoteIdList;
: Please refer to section 4.2 in page 8.

- (void)setNoteIdListSectionOwnerFromPList;
: Please refer to section 4.2 in page 8.

- (void)setPenStateWithRGB:(UInt32)color;
: Please refer to section 4.10 in page 12.

- (void)setPenThickness:(NSUInteger)thickness;
: To adjust pen thickness scale. thickness should be 1 ~ 3.

- (void)setPenStateWithPenPressure:(UInt16)penPressure;
: Please refer to section 4.5 in page 10.

- (void)setPenStateWithAutoPwrOffTime:(UInt16)autoPwrOff;
: Please refer to section 4.5 in page 10.

- (void)setPenStateAutoPower:(unsigned char)autoPower Sound:(unsigned char)sound;
: Please refer to section 4.5 in page 10.

- (void)setPenStateWithTimeTick;
: To send device system time to a pen

- (unsigned char)getPenStateWithBatteryLevel;
: deprecated

- (unsigned char)getPenStateWithMemoryUsed;
: deprecated

- (NSString *)getFWVersion;
: Please refer to section 4.7 in page 11.

- (BOOL) requestOfflineDataWithOwnerId:(UInt32)ownerId noteId:(UInt32)noteId;
: Please refer to section 4.6 in page 10.

- (void) changePasswordFrom:(NSString *)curNumber To:(NSString *)pinNumber;
: Please refer to section 4.4 in page 9.

- (void) setBTComparePassword:(NSString *)pinNumber;
: Please refer to section 4.4 in page 9.

- (void) sendUpdateFileInfoAtUrlToPen:(NSURL *)fileUrl;
: Please refer to section 4.7 in page 11.

- (void)requestWritingStartNotification;
: NJPenCommParserPageChangedNotification occurs when note or page id is changed. This method has the notification sent when the same page starts to be written again after a canvas view is closed.

- (void)resetPenRegistration;
: Please refer to section 4.1 in page 7.

- (void) connectPeripheralAt:(NSInteger)index;

: Please refer to section 4.11 in page 12.

- (void) getPenBattLevelAndMemoryUsedSize:(void (^)(unsigned char remainedBattery, unsigned char usedMemory))completionBlock;

: Please refer to section 4.14 in page 13.

7.2 NJPenCommParser.h

@protocol NJPenCommParserStrokeHandler <NSObject>
- (void) processStroke:(NSDictionary *)stroke;
- (void) activeNoteId:(int)noteId pageNum:(int)pageNumber sectionId:(int)section ownderId:(int)owner pageCreated:(NJPage *)page;
- (void) notifyPageChanging;
@optional
- (void) notifyDataUpdating:(BOOL)updating;
- (UInt32)setPenColor;
@end
: Please refer to section 4.2 in page 8.

@protocol NJPenCommParserCommandHandler <NSObject>
@optional
- (void) sendEmailWithPdf;
- (void) penConnectedByOtherApp:(BOOL)penConnected;
@end
: Please refer to section 4.12 and 4.13 in page 12 and page 13.

@protocol NJPenCommParserPasswordDelegate <NSObject>
- (void) performComparePassword:(PenPasswordRequestStruct *)request;
@end
: It is not used

@protocol NJPenCommParserStartDelegate <NSObject>
- (void) activeNoteId:(int)noteId pageNum:(int)pageNumber;
- (void) firstStrokePage: (NJPage *)fPage;
- (void) setPenCommNoteIdList;
@end
: Please refer to section 4.2 in page 8.

7.3 NJPage.h

- (UIImage *) drawStroke: (NJStroke *)stroke withImage:(UIImage *)image
size:(CGRect)bounds scale:(float)scale
offsetX:(float)offset_x offsetY:(float)offset_y
: Please refer to section 4.3 in page 9.

- END -