eBeam SmartMarker ,Smartpen Developer Guide for iOS

Luidia R&D S/W

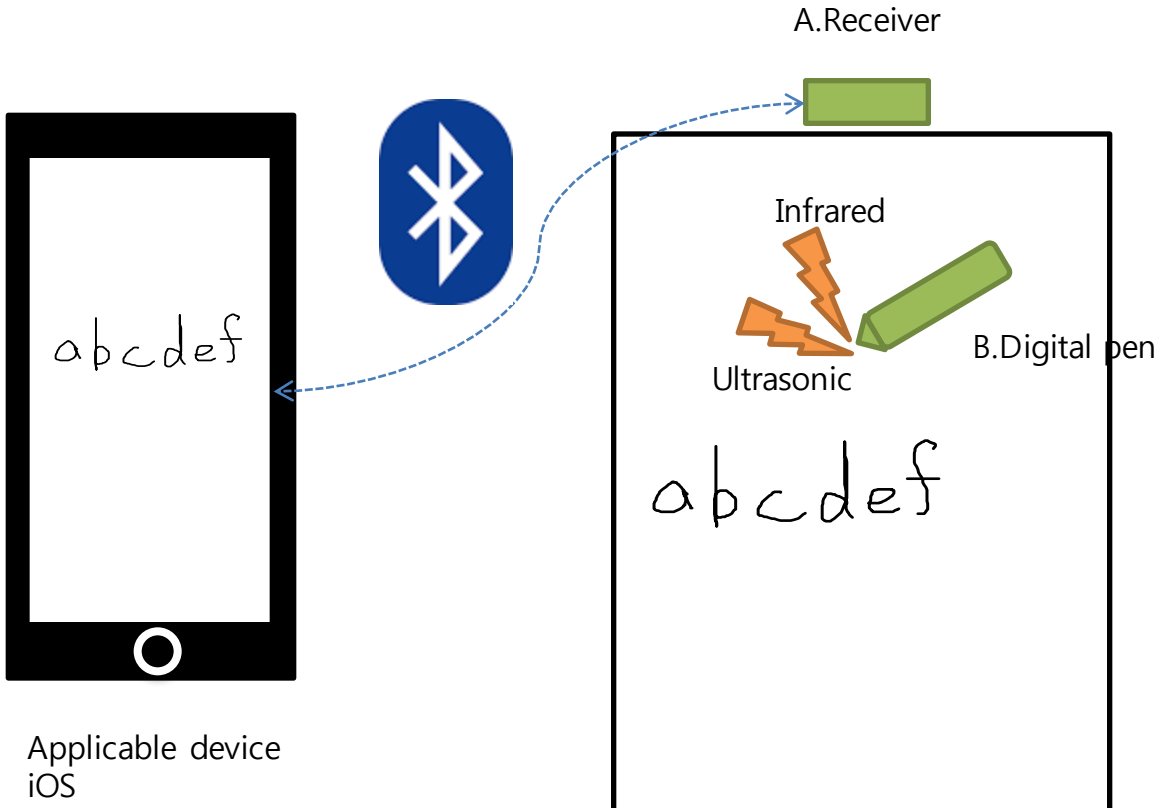
2018. 05

Contents

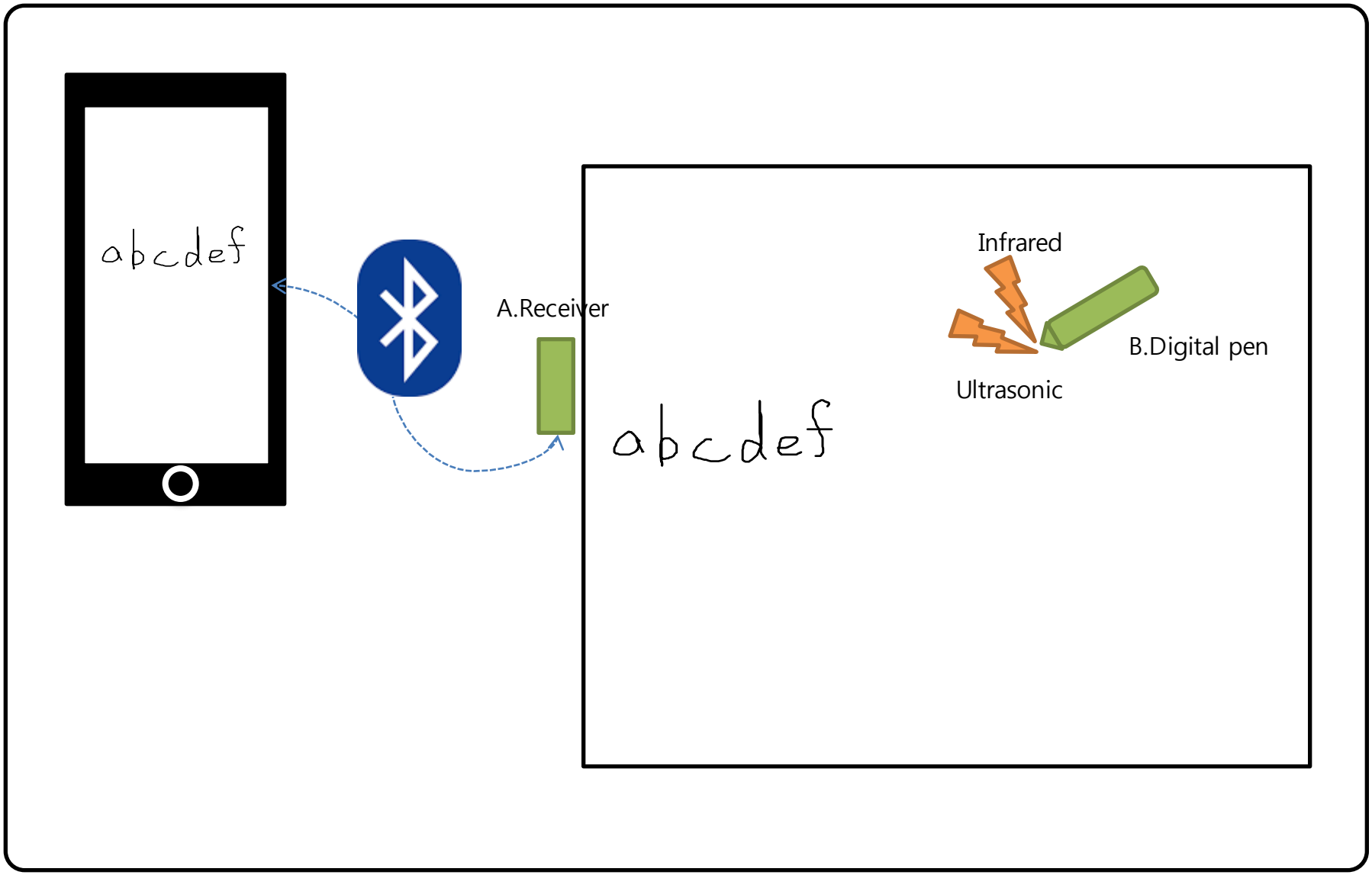
Concept > Luidia Hardwares

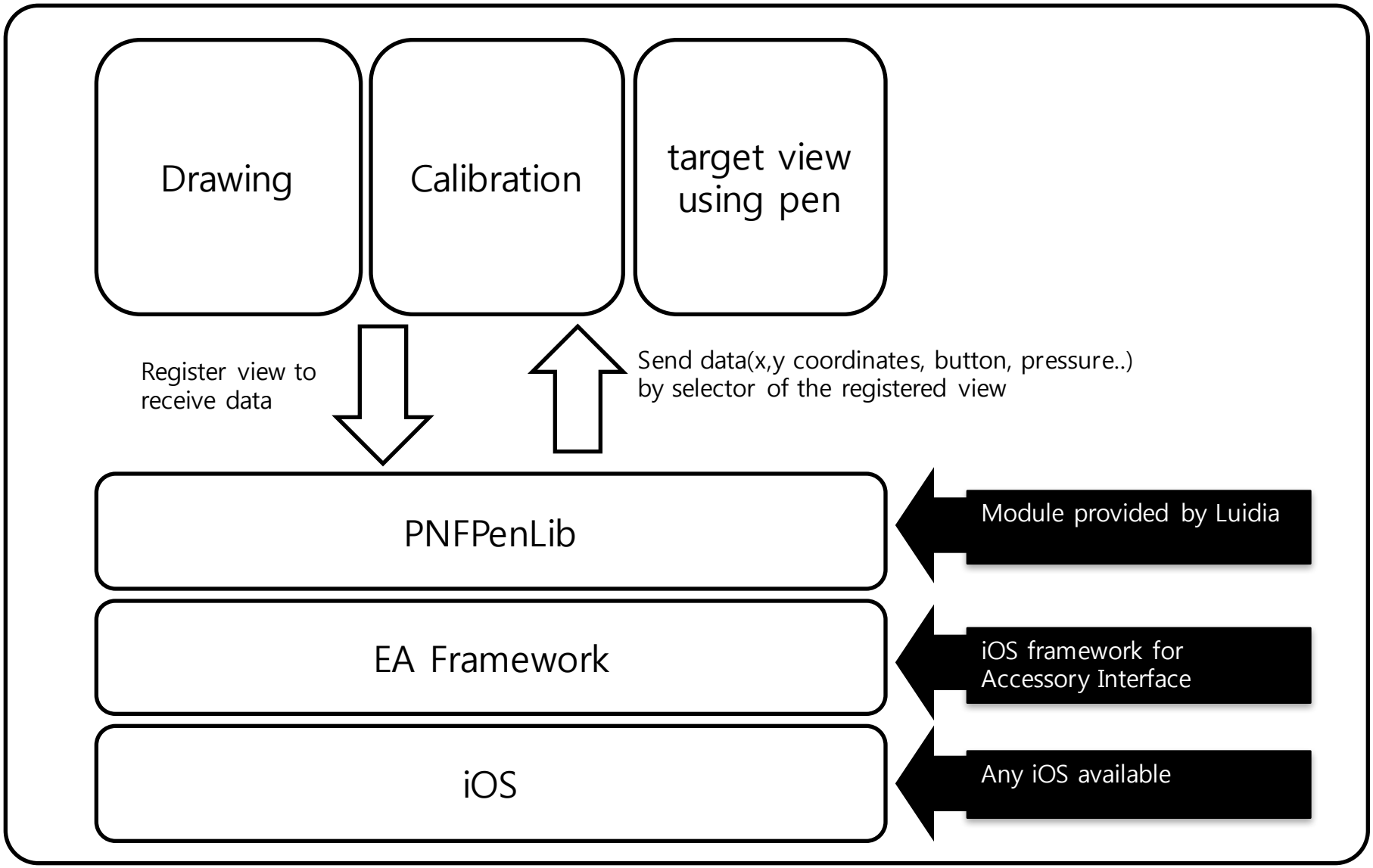
Model	Devices	Connection	Writing	Image
eBeam Smart Pen	iPhone,iPod,iPad, Windows,Android	Wireless(BLE)	On the paper Or desk	
eBeam Smart Marker	iPhone,iPod,iPad, Windows,Android	Wireless(BLE)	On the whiteboard	

Concept > Hardware Structure (eBeam SmartMarker ,Smartpen)



Concept > Hardware Structure (SmartMarker ,Smartpen)




Drawing
Calibration
target view using pen
Register view to receive data
Send data(x,y coordinates, button, pressure..) by selector of the registered view
PNFPenLib
EA Framework
iOS
Module provided by Luidia
iOS framework for Accessory Interface
Any iOS available

- Add PNFModule folder of the sample soures into your project

Development > Components of Test Sample (PenTest)

※ $(SrcHome) : [unZipped folder]/

Folder		File	Description
$(SrcHome)/PenTest/	./	main.m	
		PenTest-Info.plist	
		PenTest-Prefix.pch	
		AppDelegate.h .m	
		ViewController.h .m .xib	Main controller
		BTNameChangeViewController.m	SmartMarker ,Smartpen name change.
	DrawView/	DrawView.h .m	Drawing lines according to the coordinate from pen.
		DrawViewController.h .m .xib	
$(SrcHome)/Common/	Calibration/	PenCalibrationViewController.m .xib MarkerCalibrationViewController.m .xib	2 points calibration view(eBeam Smart Pen) 2 points calibration view(eBeam Smart Marker)
	Common/	Toast+UIView.h .m	Shows error information about Pen.
		UIImage+ImageNamed.m	Load image data
		Common.h	Default Calibration value
	PNFModule/	libPNFPenLib.a	Standard library
		PNFDefine.h	Constants
		PNFPenLib.h	Interfaces
		PNFPenLibExtension.h	Interfaces
	PNFStrokePoint/	PNFStrokePoint.h .m	Objects for drawings
	Resource/		

● PNFPenLibExtension Class

Inherits from	NSObject
Declared in	PNFPenLibExtension.h

➢ Overview

PNFPenLibExtension is the class of PNFPenLib Library to manage the information of device , make calibrated coordinates and tranfer it to the other classes.

➢ Members

ptRaw			
Type	CGPoint	Property	readonly
Description	Coordinates before calibrating		
Range	0 ~ 6500		
Device	eBeam SmartMarker ,Smartpen		
Usage			

ptConv			
Type	CGPoint	Property	readonly
Description	Calibrated coordinates		
Range	According to the target view size		
Device	eBeam SmartMarker ,Smartpen		
Usage			

PenStatus			
Type	int	Property	readonly
Description	Where pentip is pressed or not		
Range	PEN_DOWN : Pentip down PEN_MOVE : Move with Pentip down PEN_UP : Pentip up		
Device	eBeam SmartMarker ,Smartpen		
Usage			

<table>
<tr><td colspan="4">StationPosition</td></tr>
<tr><td>Type</td><td>int</td><td>Property</td><td>readonly</td></tr>
<tr><td>Description</td><td colspan="3">Current position of eBeam SmartMarker station.</td></tr>
<tr><td>Range</td><td colspan="3">DIRECTION_LEFT
DIRECTION_RIGHT
DIRECTION_TOP
DIRECTION_BOTTOM
DIRECTION_BOTH
(defined in PNFDefine.h)</td></tr>
<tr><td>Device</td><td colspan="3">eBeam SmartMarker</td></tr>
<tr><td>Usage</td><td colspan="3">[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(PenCallBackFunc:) name:@"PNF_MSG" object:nil];

....
....
-(void) PenCallBackFunc:(NSNotification *)call {
if ([szS isEqualToString:@"CHANGE_DEVECE_POSITION"] || [szS isEqualToString:@"CHANGE_DEVECE_POSITION_FIRST"]) {
if (penController.StationPosition == DIRECTION_LEFT)
position = @"Left";
else if (penController.StationPosition == DIRECTION_RIGHT)
position = @"Right";
else if (penController.StationPosition == DIRECTION_TOP)
position = @"Top";
else if (penController.StationPosition == DIRECTION_BOTTOM)
position = @"Bottom";
else
position = @"Both";
}</td></tr>
</table>

bStopped			
Type	BOOL	Property	readonly
Description	Whether Pause is set or not If it is set, Pen data is not transferred to target view.		
Range	Yes / No		
Device	eBeam SmartMarker ,Smartpen		

Usage	[penController stopPen]; // set pause
	NSLog(@"%@", penController.bStopped ? @"YES",@"NO");
	/// display YES
	
	[penController restartPen]; // release pause
	NSLog(@"%@", penController.bStopped ? @"YES",@"NO");
	/// display NO		

AudioMode			
Type	Int	Property	readonly
Description	Audio Mode of Smart Marker		
Range	YES = beep only NO = beep + voice		
Device	eBeam SmartMarker		
Usage			

Volume			
Type	Int	Property	readonly
Description	Audio volume of Smart Marker		
Range	0 ~ 255 0 = loud 255 = slient		
Device	eBeam SmartMarker		
Usage			

battery_station			
Type	Int	Property	readonly
Description	Battery status of sensor		
Range	0 ~ 100		
Device	eBeam SmartMarker ,Smartpen		
Usage			

battery_pen			
Type	Int	Property	readonly
Description	Battery status of pen		
Range	• Smart Marker 0 = High Else = Low • Smart Pen 0 ~ 100		
Device	eBeam SmartMarker ,Smartpen		
Usage			

➢ Methods

BLEInit		
Description	Start to communicate with device	
out		
input	N/A	
Device	eBeam SmartMarker ,Smartpen	

Usage	-(void) viewDidLoad
	{
	.....
	penController = [[[PNFPenLibExtension alloc] init] autorelease];
	#if TARGET_LUIDIA_EBEAMMARKER
	//[eBeam Smart marker device]
	[penController setDefaultModelCode:eBeamSmartMarker];
	[penController setProjectiveLevel:4];
	[penController fixStationPosition:DIRECTION_LEFT];
	#else
	//[eBeam Smart pen device]
	[penController setDefaultModelCode:eBeamSmartPen];
	[penController setProjectiveLevel:4];
	[penController fixStationPosition:DIRECTION_TOP];
	#endif
	[penController BLEInit];
	[penController setRetObj:self];
	[penController setRetObjForEnv:self];
	.....
	}	

BLEConnect		
Description	Connect to communicate with device	
out	int	CONNECTED : success FIRST_DATA_RECV : first data read SESSION_CLOSED: receiving error (should reconnect the device) (Define in PNFDefine.h)
input	N/A	
Device	eBeam SmartMarker ,Smartpen	
Usage	[penController BLEConnect:peripheral];	

BLEDisconnectClicked		
Description	Disconnect device	
out	Void	
input	N/A	
Device	eBeam SmartMarker ,Smartpen	
Usage	[penController BLEDisconnect];	

setRetObjForEnv		
Description	Set an object to receive the pen data for environment The object should have "-(void) PenHandlerEnv:(NSArray*)info {}"	
out	Void	
input	NSObject*	Object pointer to receive the pen alive data for environment
Device	eBeam SmartMarker ,Smartpen	

Usage	-(void) viewDidLoad
	{
	.....
	penController = [[[PNFPenLibExtension alloc] init] autorelease];
	
	[penController setRetObjForEnv:self];
	.....
	}
	-(void) PenHandlerEnv:(NSArray*)info {
	
	}	

getRetObjForEnv		
Description	Return registered object to receive pen data	
Out	NSObject*	
Input	Void	
Device	eBeam SmartMarker ,Smartpen	
Usage	[penController getRetObjForEnv];	

<table>
<tr><td colspan="3">sendCalibrationDataToDevice</td></tr>
<tr><td>Description</td><td colspan="2">Send data for calibration with position</td></tr>
<tr><td>out</td><td>Void</td><td>[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(PenCallBackFunc:) name:@"PNF_MSG" object:nil];</td></tr>
<tr><td rowspan="2">input</td><td>DEVICE_DIRECTION</td><td>position of eBeam device

DIRECTION_LEFT
DIRECTION_RIGHT
DIRECTION_TOP
DIRECTION_BOTTOM
DIRECTION_BOTH
(defined in PNFDefine.h)</td></tr>
<tr><td>CGPoint[]</td><td>Original points</td></tr>
<tr><td>Device</td><td colspan="2">eBeam SmartMarker ,Smartpen</td></tr>
<tr><td>Usage</td><td colspan="2">-(void) runApplyProcess {
// CGPoint m_CaResultPoint[4];
.....
[PenController sendCalibrationDataToDevice:(enum DEVICE_DIRECTION)type
CalibPoint:m_CalResultPoint]];
}

-(void) PenCallBackFunc:(NSNotification *) call {
NSString * szS = (NSString *) [call object];
if ([szS isEqualToString:@"CALIBRATION_SAVE_OK"]) {
.....
}else if([szS isEqualToString:@"CALIBRATION_SAVE_FAIL"] ||
[szS isEqualToString:@"DI_SEND_ERR"]){
.....
}
}</td></tr>
</table>

setCalibration		
Description	Set data for calibration	
out	Void	
input	CGRect	square which consists of calibrated coordinates
	Float	Margin between displayed point and edge of screen
Device	eBeam SmartMarker ,Smartpen	

Usage	-(void) successMarkerCalibrationViewController {
	CGSize drawingSize = [self GetDrawingSizeByCalibration];
	[penController setCalibration:
	scaleRect(CGRectMake(0, 0, drawingSize.width, drawingSize.height))
	GuideMargin:0];
	.....
	}
	
	-(void) successPenCalibrationViewController {
	CGSize drawingSize = [self GetDrawingSizeByCalibration];
	[penController setCalibration:
	scaleRect(CGRectMake(0, 0, drawingSize.width, drawingSize.height))
	GuideMargin:0];
	}	

setProjectiveLevel		
Description	Set calibration points	
out	Void	
input	Int	
Device	eBeam SmartMarker ,Smartpen	

Usage	-(void) viewDidLoad
	{
	.....
	[penController setProjectiveLevel:4];
	.....
	
	}	

<table>
<tr><td colspan="3">changeDeviceName</td></tr>
<tr><td>Description</td><td colspan="2">Send change name data for SmartMarker ,Smartpen</td></tr>
<tr><td>out</td><td>Void</td><td>[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(PenCallBackFunc:) name:@"PNF_MSG" object:nil];</td></tr>
<tr><td rowspan="2">input</td><td>NSString</td><td>deviceName</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Device</td><td colspan="2">eBeam SmartMarker ,Smartpen</td></tr>
<tr><td>Usage</td><td colspan="2">-(IBAction)changeClicked:(id)sender {
.....
[m_PenController changeDeviceName:changeName];
}

-(void) PenCallBackFunc:(NSNotification *) call {
NSString * szS = (NSString *) [call object];
if ([szS isEqualToString:@"ChangeDeviceName_OK"]) {
.....
}else if([szS isEqualToString:@"ChangeDeviceName_FAIL"] ||
[szS isEqualToString:@"DI_SEND_ERR"]) {
.....
}
}</td></tr>
</table>

changeAudioMode		
Description	Change Audio mode of Smart Marker	
Out	Void	
Input	BOOL	Yes:/No
Device	eBeam SmartMarker	
Usage	[penController changeAudioMode:YES]; -> Change to beep only [penController changeAudioMode:NO]; -> change to beep and voice	

changeVolume		
Description	Change audio volume	
Out	Void	
Input	int	0 ~ 255
Device	eBeam SmartMarker	
Usage	[penController changeVolume:0]; -> max [penController changeVolume:255]; -> min	

ReadQ		
Description	Read one data from read Queue	
Out	NSDictionary	
Input	Void	
Device	eBeam SmartMarker ,Smartpen	

Usage	NSDictionary* dic = [penController ReadQ];
	CGPoint ptRaw = [[dic objectForKey:@"ptRaw"] CGPointValue];
	CGPoint ptConv = [[dic objectForKey:@"ptConv"] CGPointValue];
	int PenStatus =[[dic objectForKey:@"PenStatus"] intValue];
	int Temperature = [[dic objectForKey:@"Temperature"] intValue];
	int modelCode = [[dic objectForKey:@"modelCode"] intValue];
	int SMPenFlag = [[dic objectForKey:@"SMPenFlag"] intValue];
	int SMPenState = [[dic objectForKey:@"SMPenState"] intValue];
	int pressure = [[dic objectForKey:@"pressure"] intValue];	

RemoveQ		
Description	Delete one data from read Queue	
Out	Void	
Input	Void	
Device	eBeam SmartMarker ,Smartpen	
Usage	[penController removeQ];	

ClearQ		
Description	Clear all data from read Queue	
Out	Void	
Input	Void	
Device	eBeam SmartMarker ,Smartpen	
Usage	[penController ClearQ];	

StartReadQ		
Description	Read Pen mode through Read Queue	
Out	Void	
Input	Void	
Device	eBeam SmartMarker ,Smartpen	

Usage	[penController StartReadQ];
	.....
	-(void) runReadThread {
	@autoreleasepool {
	while (1) {
	if (readThreadStop) {
	break;
	}
	
	if ([[UIApplication sharedApplication] isIgnoringInteractionEvents]) {
	[NSThread sleepForTimeInterval:0.02];
	continue;
	}
	
	NSDictionary* dic = [penController ReadQ];
	if(dic) {
	[self performSelectorOnMainThread:@selector(PenHandlerWithDictionary:) withObject:dic waitUntilDone:YES];
	[penController RemoveQ];
	}
	else {
	[NSThread sleepForTimeInterval:0.02];
	}
	} // while (1) {
	}
	}	

EndReadQ		
Description	Read Pen mode through Notification	
Out	Void	
Input	Void	
Device	eBeam SmartMarker ,Smartpen	

Usage	(void) viewDidLoad
	{
	[[NSNotificationCenter defaultCenter] addObserver:self
	selector:@selector(PenHandlerWithMsg:)
	name:@"PNF_PEN_READ_DATA"
	object:nil];
	}
	-(void) PenHandlerWithMsg:(NSNotification*) note {
	NSDictionary* dic = [note object];
	if ([penController getRetObjForEnv] != self) return;
	[self PenHandlerWithDictionary:dic];
	}
	-(void) PenHandlerWithDictionary:(NSDictionary*) dic {
	int PenStatus = [[dic objectForKey:@"PenStatus"] intValue];
	CGPoint ptRaw = [[dic objectForKey:@"ptRaw"] CGPointValue];
	CGPoint ptConv = [[dic objectForKey:@"ptConv"] CGPointValue];
	int Temperature = [[dic objectForKey:@"Temperature"] intValue];
	int modelCode = [[dic objectForKey:@"modelCode"] intValue];
	int SMPenFlag = [[dic objectForKey:@"SMPenFlag"] intValue];
	int SMPenState = [[dic objectForKey:@"SMPenState"] intValue];
	int press = [[dic objectForKey:@"pressure"] intValue];
	[self PenHandlerWithArgs:ptRaw
	ptConv:ptConv
	PenStatus:PenStatus
	Temperature:Temperature
	ModelCode:modelCode
	SMPenFlag:SMPenFlag
	SMPenState:SMPenState
	Pressure:press];
	}	

➢ Overview

Create and initialize object PNFPenLibExtension

➢ Example

```
1.      Create PNFPenLibExtension object
        penController = [[PNFPenLibExtension alloc] init];

2.      Appoint the calibration points
#if TARGET_LUIDIA_EBEAMMARKER
        [penController setDefaultModelCode:eBeamSmartMarker];
        [penController setProjectiveLevel:4];          //4 points
        [penController fixStationPosition:DIRECTION_LEFT];
#else
        [penController setDefaultModelCode:eBeamSmartPen];
        [penController setProjectiveLevel:4];          //4 points
        [penController fixStationPosition:DIRECTION_TOP];
#endif
        [m_PenController BLEInit];

3.      Set object to receive data
          [penController setRetObjForEnv:self];
```

example source: ViewController.h ViewController.m

➢ Overview

Internally PNFPenController is supposed to call selector named as "PenHandlerWithMsg" of object set by "PNF_PEN_READ_DATA" whenever the pen moves.

➢ Example

```
// if [penController StartReadQ];
-(void) ReadThreadStart {
    [self addDebugText:@"ReadThreadStart"];
    if (readThread == nil) {
        readThread = [[NSThread alloc] initWithTarget:self
                                        selector:@selector(runReadThread) object:self];
        readThreadStop=NO;
        readThreadPause=NO;
        [readThread start];

    }
    if (penController) {
        [penController StartReadQ];
    }
}

// if [penController EndReadQ];
-(void) PenHandlerWithMsg:(NSNotification*) note {
    NSDictionary* dic = [note object];
    if ([penController getRetObj] != self)
        return;
    [self PenHandlerWithDictionary:dic];
}
```

example source: ViewController.h ViewController.m

➢ Example

```
// if [penController StartReadQ];
-(void) runReadThread {
    @autoreleasepool {
        while (1) {
            if (readThreadStop) {
                break;
            }

            NSDictionary* dic = [penController ReadQ];
            if(dic == nil) {
                [self performSelector:@selector(sleepThread:)  onThread:readThread withObject:[NSNumber numberWithFloat:0.02] waitUntilDone:YES];
            }else {
                [self performSelectorOnMainThread:@selector(PenHandlerWithDictionary:)  withObject:dic  waitUntilDone:YES];
                [penController RemoveQ];
            }
        }
    }
}

// if [penController StartReadQ];
-(void) ReadThreadOff {
    [self addDebugText:@"ReadThreadOff"];
    readThreadStop = YES;
    [NSThread sleepForTimeInterval:0.2];
    if (readThread) {
        [readThread cancel];
        [readThread release];
        readThread = nil;
    }
    if (penController) {
        [penController EndReadQ];
    }
}
```

example source: ViewController.h ViewController.m

➢ Example

```
-(void) PenHandlerWithDictionary:(NSDictionary*)  dic {
    int PenStatus   = [[dic objectForKey:@"PenStatus"]  intValue];
    CGPoint ptRaw  = [[dic objectForKey:@"ptRaw"]  CGPointValue];
    CGPoint ptConv = [[dic objectForKey:@"ptConv"] CGPointValue];
    int Temperature  = [[dic objectForKey:@"Temperature"]  intValue];
    int modelCode  = [[dic objectForKey:@"modelCode"]  intValue];
    int SMPenFlag  = [[dic objectForKey:@"SMPenFlag"]  intValue];
    int SMPenState  = [[dic objectForKey:@"SMPenState"]   intValue];
    int press = [[dic objectForKey:@"pressure"] intValue];
    [self  PenHandlerWithArgs:ptRaw
                       ptConv:ptConv
                    PenStatus:PenStatus
                  Temperature:Temperature
                    ModelCode:modelCode
                    SMPenFlag:SMPenFlag
                   SMPenState:SMPenState
                     Pressure:press];
}
-(void) PenHandlerWithArgs:(CGPoint)  Arg_ptRaw ptConv:(CGPoint)  Arg_ptConv PenStatus:(int)  Arg_PenStatus
                Temperature:(int)  Arg_Temperature ModelCode:(int)   Arg_modelCode
                 SMPenFlag  :(int) Arg_SMPenFlag SMPenState:(int)   Arg_SMPenState
                   Pressure:(int)  Arg_pressure {
            CGPoint ptDrawing;
            switch (Arg_PenStatus) {
                    case PEN_DOWN:
                        break;
                    case PEN_MOVE:
                        break;
                    case PEN_UP:
                        break;
                }
            }
            ptDrawing  = PenController.ptConv ;
}
```

example source: ViewController.h ViewController.m

➢ Overview

Information of device status is sent by notification named as "PNF_LOG_MSG".

➢ Example

1. Add Notification

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(FreeLogMsg:)
name:@"PNF_LOG_MSG" object:nil];
```

2. Handler for Message

```
-(void) FreeLogMsg:(NSNotification *) note
{
    NSString * szS = (NSString *) [note object];
    if ([szS isEqualToString :@"FAIL_LISTENING"] ) {
    }
    else if ([szS isEqualToString:@"CONNECTED"]) {
    }
    else if ([szS isEqualToString:@"INVALID_PROTOCOL"]) {
    }
    else if ([szS isEqualToString:@"SESSION_CLOSED"]) {
    }
    else if ([szS isEqualToString:@"PEN_RMD_ERROR"]) {

    }
    else if ([szS isEqualToString:@"FIRST_DATA_RECV"]) {
    }
}
```

Log String Message	Description
CONNECTED	Device is connected
NOT_CONNECTED	Device is disconnected
FAIL_LISTENING	Fail to receive. Need to reconnect.
INVALID_PROTOCOL	Invalid hardware
SESSION_CLOSED	Session is disconnected
FIRST_DATA_RECV	First data is received after connecting
PEN_RMD_ERROR	Abnormal drawing data

example source : ViewController.h ViewController.m

➢ Overview

Information of device status is sent by notification named as "PNF_MSG".

➢ Example

1. Add Notification

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(PenCallBackFunc:)
name:@"PNF_MSG" object:nil];
```

2. Handler for Message

```
-(void) PenCallBackFunc:(NSNotification *)call {
    NSString * szS = (NSString *) [call object];
    if([szS isEqualToString:@"BATTERY_INFO"]) {
    }
    else if([szS isEqualToString:@"NEW_PAGE"]) {
    }
    else if([szS isEqualToString:@"DUPLICATE_PAGE"]) {
    }
    else if ([szS isEqualToString:@"CHANGE_DEVECE_POSITION"]) {
    }
    else if([szS isEqualToString:@"CHANGE_DEVECE_POSITION_FIRST"]) {
    }
    else if([szS isEqualToString:@"DI_SEND_ERR"]) {
    }
    else if([szS isEqualToString:@"ChangeDeviceName_OK"]) {
    }
    else if([szS isEqualToString:@"ChangeDeviceName_FAIL"]) {
    }
    else if([szS isEqualToString:@"CALIBRATION_SAVE_OK"]) {
    }
    else if([szS isEqualToString:@"CALIBRATION_SAVE_FAIL"]) {
    }
}
```

Log String Message	Description
BATTERY_INFO	Battery information
NEW_PAGE	Button smartMarker ,Smartpen
DUPLICATE_PAGE	Long press button smartMarker ,Smartpen
CHANGE_DEVECE_POSITION	Change device position
CHANGE_DEVECE_POSITION_FIRST	Change device position first
DI_SEND_ERR	Send data fail
ChangeDeviceName_OK	Device name change success
ChangeDeviceName_FAIL	Device name change fail
CALIBRATION_SAVE_OK	Calibration change success
CALIBRATION_SAVE_FAIL	Calibration change fail

example source : ViewController.h ViewController.m

➢ Overview

Pen coordinates is converted to screen coordinates by projective matrix which is set in the calibration view.

➢ Example

```
 1. create calibration controller
if (penController.modelCode == eBeamSmartMarker) {
        MarkerCalibrationViewController* cVController = [[MarkerCalibrationViewController alloc]
        initWithNibName:@"MarkerCalibrationViewController" bundle:nil];
} else {
       PenCalibrationViewController* cVController = [[PenCalibrationViewController alloc]
        initWithNibName:@"PenCalibrationViewController" bundle:nil];
}

 2. connect Pen controller and calibration controller
    [cVController SetPenController:penController];

 3. set calibration controller as target view
    [PenController setRetObj:cVController];

 4. show calibration view
     [self presentViewController:cVController animated:NO completion:^{ }];
```

example source : CalibViewController.h CalibViewController.m

➢ Overview

Transfers coordinate data to the SmartMarker,Smartpen.

➢ Example

```
5. send calibration data
    CGPoint  m_CaResultPoint[4];
    ……
    [PenController sendCalibrationDataToDevice:(enum  DEVICE_DIRECTION)type CalibPoint:m_CalResultPoint]];

6. Receive callback calibration data
    -(void) PenCallBackFunc:(NSNotification *) call {
        NSString * szS = (NSString *) [call object];
        if ([szS isEqualToString:@"CALIBRATION_SAVE_OK"]) {
            ……
        }else if ([szS isEqualToString:@"CALIBRATION_SAVE_FAIL"] || [szS isEqualToString:@"DI_SEND_ERR"]) {
            ……
        }
    }
```

example source : CalibViewController.h CalibViewController.m

➢ Overview

Calibration data is saved automatically by this library.
App need not save the data.

➢ Example

```
7. Save calibration data
    /// after click the last calibration point
    CGSize drawingSize = [self GetDrawingSizeByCalibration];
    [penController setCalibration:
                        scaleRect(CGRectMake(0, 0, drawingSize.width, drawingSize.height))
                        GuideMargin:0];
```

example source : CalibViewController.h CalibViewController.m