Mobilogics mobilising your business

Mobilogics
Barcode 4.1

Thu Mar 14 2013 12:40:05

Chapter 1

Mobilogics Barcode SDK Manual

Introduction of SDK

Welcome to use Mobilogics Software Development Kit (SDK). SDK enables you to create applications that run on Mobilogics Products (iPDT380, iScan and aScan). This technology lets you build a single binary that takes you advantage of new features when running on a system that supports them.

You can find out SDK Release Notes here.

For more information please visit our web site: http://www.mobilogics.com.tw

How to use

Add Barcode Framework to your Project

Use Barcode Framework in your project, just follow these three steps:

1. Add libstdc++.dylib to your project.
2. Add ExternalAccessory framework to your project.
3. Add Barcode framework and MobilogicsCore framework to your project.
   You must install Mobilogics SDK first, after you installed, you can found Barcode framework and Mobilogics-Core framework in

   ```
   /Library/Frameworks/Mobilogics.framework/Frameworks
   ```

4. Add

   ```
   #import <Barcode/Framework.h>
   ```

   to your .pch header file.
5. Add link options to your project's **Other Linker Flags**

   ```
   -ObjC
   -all_load
   ```

6. If you using SDK with Xcode 4.0 or laster version.
   You need to add array item which named [Supported external accessory protocols]
   in your Info.plist. And you need add three string item in this array as fllow:

Item 0: tw.com.mobilogics.iPDT380

Item 1: tw.com.mobilogics.iscan

Item 2: tw.com.mobilogics.ascan

Item 3: tw.com.mobilogics.AK110

Item 4: tw.com.mobilogics.ipdt5

Initialize Barcode Framework

Before you use Barcode Framework, you must initialize framework as follow code :

```
[[MLScanner sharedInstance] setup];
```

In BarcodeExample with ARC enable, we put this code on this method like this:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
      (NSDictionary *)launchOptions {

  [[MLScanner sharedInstance] setup];

  [self.window makeKeyAndVisible];

  return YES;
}
```

Connect & Disconnect Notification

After you initialize Barcode Framework, you will got notification when accessory connected or disconnected by follow source code:

- add follow source code to got notification from SDK:

```
  [[MLScanner sharedInstance] addAccessoryDidConnectNotification:self]
      ;
  [[MLScanner sharedInstance] addAccessoryDidDisconnectNotification:
      self];
  [[MLScanner sharedInstance] addReceiveCommandHandler:self];

In BarcodeExample, we put this code on this method like this:

- (void)viewDidLoad {
  [super viewDidLoad];

  [[MLScanner sharedInstance] addAccessoryDidConnectNotification:self]
      ;
  [[MLScanner sharedInstance] addAccessoryDidDisconnectNotification:
      self];
  [[MLScanner sharedInstance] addReceiveCommandHandler:self];
}

And you must implement <NotificationHandler> protocol with two methods to handle notification:

-(void)connectNotify;    // When you got connected notification, framework will
       callback here.
-(void)disconnectNotify; // When you got disconnected notification, framework
       will callback here.
```

- If you want to check connection established, you can use follow source code:

```
  [[MLScanner sharedInstance] isConnected]

You can read isConnected method on MLConnection class.\n\r
In BarcodeExample, we also use follow source code to determent connected or not:

  if ([[MLScanner sharedInstance] isConnected]) {
                ...
  } else {
                ...
  }
```


Scan

After barcode scanner already connected, you can use follow source code to scan barcode:

```
  [[MLScanner sharedInstance] scan];
```

We suggested that detect accessory connected or not. If accessory disconnected then ScanShot command will be ignore.

In BarcodeExample, we put this code on this method like this:

```
- (IBAction)scan:(id)sender {
  [[MLScanner sharedInstance] scan];
}
```

Scan and Receive Barcode String are Asynchronies and independent. So you will receive barcode string not just execute ScanShot command, accessory hardware triggered scan will return barcode string too.

Receive Barcode

If you want to receive barcode string, you must implement ReceiveCommandHandler protocol. It contains two methods. For more details please reference ReceiveCommandHandler protocol.

```
- (BOOL)isHandler:(NSObject<ReceiveCommandProtocol> *)command;
- (void)handleRequest:(NSObject<ReceiveCommandProtocol> *)command;
```

In BarcodeExample, we use follow source code to handle or not barcode string:

```
- (BOOL)isHandler:(NSObject<ReceiveCommandProtocol> *)command {
  // we just handle the class which extensions ReceiveCommand class.
  if ([command isKindOfClass:[ReceiveCommand class]]) {
    return TRUE;
  }
  return FALSE;
}

- (void)handleRequest:(NSObject<ReceiveCommandProtocol> *)command {
  [label setText:[command getReceiveString]];
}
```

Supported Mobilogics Accessory

1. iPDT380
2. iScan
3. aScan
4. AK110
5. iPDT5

Chapter 2

Release Notes

Mobilogics SDK Release Notes

Version 4.1 (2013-03-14)

- Fixed bug of switch App from active to background and from background to active many times between in very tiny time will cause App crash because of iOS multithreads race condition error.
- Fixed bug of hardware trigger many times in very tiny time will cause call getReceiveString got long string error.
- Support AK110.
- Support iPDT5
- Replace MLConnection with new easy to use class: MLScanner.
- Fixed Command Layer architecture to reduce memory used and speed up.
- Fixed and reviewed all model commands.
- Add some model's battery commands.

Version 4.0 (2012-10-19)

- Support iPhone 5 armv7s platform.
- Drop support armv6 or above platform.

Version 3.2 (2012-07-12)

- Support aScan.
- Support multi-accessory.
- Support iOS 6 beta 2 and later.
- Support Xcode 4.4 on OS X Lion.
- Support Xcode 4.5 beta on OS X Mountain Lion.
- Support ARC.
- Still support both Armv6 and Armv7 platform from iOS 4.x $\sim$ 5.x
- Add Full SDK Manual within installed package.
- Remove Project Templates.

Version 2.4 (2012-04-25)

- Fixed compatible with Apple MessageUI framework and change Connection to MLConnection.
- Fixed some minor issue on BarcodeExample.
- Changed release package mode.

Version 2.3 (2012-03-16)

- Fixed some minor bug on iOS5.x sleep mode.
- Support both Armv6 and Armv7 platform from iOS 4.x $\sim$ 5.x.

Version 2.2 (2011-12-22)

- Fixed Barcode Example source code can't work on iOS 5.0.1.
- Fixed iScan on iPad2 iOS 5.x will return empty string

Version 2.1 (2011-10-07)

- Fixed Barcode Framework CPU & Memory extra loading bug.

Version 2.0.2 (2011-10-05)

- Fixed BarcodeExample source code can't running bug.

Version 2.0.1 (2011-07-08)

- Fixed Connection object's behavior to balance cpu loading.
- Clone Connection class to MBConnection class to avoid some other framework had the same name class issue.

Version 2.0 (2011-05-09)

- Fixed Connection connectedAccessory method bug.
- Support Xcode 4.x (include Xcode 4.1 preview 4).

Version 1.5 (2011-04-18)

- Fixed iPad/iPhone/iPod connection compatible bug.
- Add more compatible with iOS 4.3.x.

Version 1.4 (2011-03-31)

- Add Accessory connection sensitive.

Version 1.3 (2011-03-17)

- Reduce Barcode Framework.
- Restructure Barcode Framework.
- Add External Accessory to DevTool Framework.

Version 1.2 (2011-02-19)

- Fixed minor bugs.

Version 1.1 (2011-02-18)

- SDK Lite add Barcode Example.
- Fixed SDK install scripts error.
- Support iOS v4.3 beta.

Version 1.0 (2011-02-16)

- Fixed minor bugs.

Version 1.0RC2 (2011-01-21)

- Add Barcode Example template.
- Fixed .pch header import setting.

Version 1.0RC1 (2011-01-21)

- Replace NSLog with LogError, LogWarn, LogInfo, LogVerbose functions.
- Fixed Navigation-based template fit new functions.
- Fixed DevTool framework header file to fit template.
- Fixed Barcode framework header file to fit template.
- Add OpenGL ES Application template support.
- Add Split View-based template support.
- Add Tab Bar Application template support.
- Add Utility Application template support.
- Add View-based Application template support.
- Add Window-based Application template support.

Version 0.9a.4 (2011-01-07)

- Fixed minor bugs.

Version 0.9a.3 (2011-01-07)

- Fixed frameowrk to support arm6 platform.

Version 0.9a.2 (2011-01-04)

- Fixed minor bugs.

Version 0.9a.1 (2011-01-03)

- Fixed Navigation-based Application template about within build Unit Test error.

Version 0.9a (2010-12-30)

- Add GHUnit framework to replace GTM Unit Test framework.
- Barcode framework change to new framework release type.
- DevTool framework change to new framework release type.
- Change Navigation-based Application template to fit all frameworks.

Version 0.9 (2010-12-28)

- Fixed Unit Test in iOS 4.2.1 compile error. After this version developer should compile in simulator mode before compile in device mode, because Unit Test can't show assert error in compile in device mode.
- Add MLog to replace NSLog.
- Fixed Navigation-based Application template error.
- Some other template in this version will not work completely, will fixed it next release.

Chapter 3

License

Mobilogics

Mobilogics SDK SOFTWARE LICENSE AGREEMENT

PLEASE READ THIS SOFTWARE LICENSE AGREEMENT ("LICENSE") CAREFULLY BEFORE USING Mobilogics SDK. BY USING Mobilogics SDK AS APPLICABLE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, DO NOT USE THE Mobilogics SDK.

Intellectual Property

As between you and Mobilogics, you agree and acknowledge that Mobilogics owns all rights, title and interest in Mobilogics SDK, including without limitation all associated Intellectual Property Rights. "Intellectual Property Rights" means any and all rights existing from time to time under patent law, copyright law, trade secret law, trademark law, unfair competition law, and any and all other proprietary rights, and any and all applications, renewals, extensions and restorations thereof, now or hereafter in force and effect worldwide. You agree to not remove, obscure, or alter Mobilogicss or any third party's copyright notice, trademarks, or other proprietary rights notices affixed to or contained within or accessed in conjunction with or through Mobilogics SDK.

You agree that the Mobilogics SDK contain proprietary content, information and material that is owned by Mobilogics and/or its licensors, and is protected by applicable intellectual property and other laws, including but not limited to copyright. You agree that you will not use such proprietary content, information or materials other than for permitted use of Mobilogics SDK or in any manner that is inconsistent with the terms of this License or that infringes any intellectual property rights of a third party or Mobilogics.

Limitation of Liability.

TO THE EXTENT NOT PROHIBITED BY APPLICABLE LAW, IN NO EVENT SHALL Mobilogics BE LIABLE FOR PERSONAL INJURY, OR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, CORRUPTION OR LOSS OF DATA, FAILURE TO TRANSMIT OR RECEIVE ANY DATA, BUSINESS INTERRUPTION OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES, ARISING OUT OF OR RELATED TO YOUR USE OF OR INABILITY TO USE THE Mobilogics SDK AND SERVICES OR ANY THIRD PARTY SOFTWARE OR APPLICATIONS IN CONJUNCTION WITH THE Mobilogics SDK, HOWEVER CAUSED, REGARDLESS OF THE THEORY OF LIABILITY (CONTRACT, TORT OR OTHERWISE) AND EVEN IF Mobilogics HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF LIABILITY FOR PERSONAL INJURY, OR OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS LIMITATION MAY NOT APPLY TO YOU.

Chapter 4

Class Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<CommandProtocol> . . . 17
Command . . . 17
MLAccessory . . . 18
AScan . . . 17
IPDT380 . . . 18
IScan . . . 18
MLScanner . . . 19
<NotificationHandler> . . . 23
<ReceiveCommandHandler> . . . 24
ScanShot . . . 25

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AScan
: AScan is used to abstract Mobilogics aScan accessory in framework 17

Command
: A simple implement of command layer object 17

<CommandProtocol>
: Command layer objects must implement this protocol 17

IPDT380
: IPDT380 is used to abstract Mobilogics iPDT380 accessory in framework 18

IScan
: IScan is used to abstract Mobilogics iScan accessory in framework 18

MLAccessory
: Accessory is used to abstract all Mobilogics accessory in framework 18

MLScanner
: Create and maintain accessory connection. After you had connection, you can scan barcode through iPDT380, iScan or AScan 19

<NotificationHandler>
: IPDT380, iScan or aScan connected or disconnected event handler must implement this protocol 23

<ReceiveCommandHandler>
: Recevie command handler must implement this protocol 24

ScanShot
: A command to trigger iPDT380, iScan or aScan to scan barcode 25

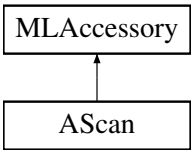
Chapter 6

Class Documentation

6.1 AScan Class Reference

AScan is used to abstract Mobilogics aScan accessory in framework.

Inheritance diagram for AScan:



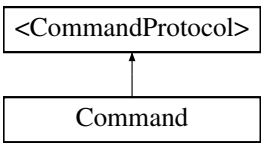
6.1.1 Detailed Description

AScan is used to abstract Mobilogics aScan accessory in framework.

6.2 Command Class Reference

A simple implement of command layer object.

Inheritance diagram for Command:



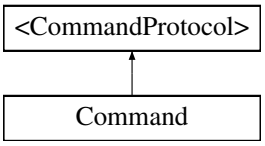
6.2.1 Detailed Description

A simple implement of command layer object.

6.3 <CommandProtocol> Protocol Reference

Command layer objects must implement this protocol.

Inheritance diagram for <CommandProtocol>:



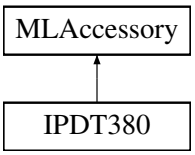
6.3.1 Detailed Description

Command layer objects must implement this protocol.

6.4 IPDT380 Class Reference

IPDT380 is used to abstract Mobilogics iPDT380 accessory in framework.

Inheritance diagram for IPDT380:



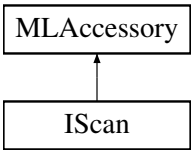
6.4.1 Detailed Description

IPDT380 is used to abstract Mobilogics iPDT380 accessory in framework.

6.5 IScan Class Reference

IScan is used to abstract Mobilogics iScan accessory in framework.

Inheritance diagram for IScan:



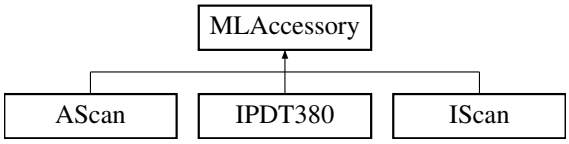
6.5.1 Detailed Description

IScan is used to abstract Mobilogics iScan accessory in framework.

6.6 MLAccessory Class Reference

Accessory is used to abstract all Mobilogics accessory in framework.

Inheritance diagram for MLAccessory:



6.6.1 Detailed Description

Accessory is used to abstract all Mobilogics accessory in framework.

6.7 MLScanner Class Reference

Create and maintain accessory connection. After you had connection, you can scan barcode through iPDT380, iScan or AScan.

Public Member Functions

- (void) - setup
- (NSArray ∗) - connectedAccessory
- (BOOL) - isConnected
- (void) - execute:
- (void) - addAccessoryDidConnectNotification:
- (void) - removeAccessoryDidConnectNotification:
- (void) - addAccessoryDidDisconnectNotification:
- (void) - removeAccessoryDidDisconnectNotification:
- (void) - addReceiveCommandHandler:
- (void) - removeReceiveCommandHandler:
- (void) - scan
- (void) - batteryRemain
- (void) - chargeBattery
- (void) - vibraMotorStrength:
- (void) - beepSwitch:
- (BOOL) - interfaceOrientationNeedUpsideDown
- (NSNumber ∗) - batteryCapacity

Static Public Member Functions

- (id) + sharedInstance

6.7.1 Detailed Description

Create and maintain accessory connection. After you had connection, you can scan barcode through iPDT380, iScan or AScan.

Connection is an Translate Layer, it can used to create, maintain connection and execute commands which is define in Command Layer.

6.7.2 Member Function Documentation

6.7.2.1 - (void) addAccessoryDidConnectNotification: (NSObject <NotificationHandler> *) *handler*

Add handler to handle iPDT380, iScan or aScan connected event. Handler must implement NotificationHandler protocol

If you use UIView, you can add handler in viewDidLoad. Please remove handler in viewDidUnload.

See also

- removeAccessoryDidConnectNotification: handler
- addAccessoryDidDisconnectNotification: handler
- removeAccessoryDidDisconnectNotification: handler
NotificationHandler

Parameters

handler	The object implement NotificationHandler protocol which can handle accessory connected notification.

6.7.2.2 - (void) addAccessoryDidDisconnectNotification: (NSObject <NotificationHandler> *) *handler*

Add handler to handle iPDT380, iScan or aScan disconnected event. Handler must implement NotificationHandler protocol

If you use UIView, you can add handler in viewDidLoad. Please remove handler in viewDidUnload.

See also

- addAccessoryDidConnectNotification: handler
- removeAccessoryDidConnectNotification: handler
- removeAccessoryDidDisconnectNotification: handler
NotificationHandler

Parameters

handler	The object implement NotificationHandler protocol which can handle accessory disconnected notification.

6.7.2.3 - (void) addReceiveCommandHandler: (NSObject <ReceiveCommandHandler> *) *handler*

Add handler to handle receive command layer object which receive from iPDT380, iScan or aScan. Send Command and Receive Command is Asynchronous, so after you send command, you can fire and forget.

Handler must implement ReceiveCommandHandler protocol

If you use UIView, you can add handler in viewDidLoad. Please remove handler in viewDidUnload.

See also

- removeReceiveCommandHandler: handler
ReceiveCommandHandler

Parameters

handler	The object implement ReceiveCommandHandler protocol which can handle receive command.

6.7.2.4 - (NSNumber *) batteryCapacity

get battery capacity. before you get this information, you must call batteryRemain function first.

Return values

NSNumber	return battery capacity information with percentage number, over 100% means in charging. For example: 12.5 means 12.5%, 110% means in charging.

6.7.2.5 - (void) batteryRemain

Get scanner accessory battery info

6.7.2.6 - (void) beepSwitch: (BOOL) *status*

switch scanner accessory beep on or off

Parameters

status	TRUE=on, FALSE=off

6.7.2.7 - (void) chargeBattery

set scanner accessory battery charge

6.7.2.8 - (NSArray *) connectedAccessory

Get the connected accessories array.

Return values

*NSArray**	The Array include connected accessories which you supported.

6.7.2.9 - (void) execute: (NSObject <**CommandProtocol**> *) *command*

Execute command.

Send command layer object to iPDT380, iScan and aScan, but when isConnected return FALSE, command will be ignore.

See also

- isConnected
ScanShot

Parameters

command	The command implement NSObject<CommandProtocol> in command layer that you want to send to scanner accessory.

6.7.2.10 - (BOOL) interfaceOrientationNeedUpsideDown

detect app layout need upside down or not

Return values

BOOL	if app layout need upsidedown return TRUE, else return FALSE

6.7.2.11 - (BOOL) isConnected

Detect scanner accessory is connected or not.

Return values

TRUE	If one of iPDT380, iScan or aScan is connected and work fine.
FALSE	If nor iPDT380, iScan or aScan is not connected, or device error. In this status, execute command will be ignore.

6.7.2.12 - (void) removeAccessoryDidConnectNotification: (NSObject <NotificationHandler> *) *handler*

Remove handler not to handle iPDT380, iScan or aScan connected event After you remove handler, you will not trigger with iPDT380, iScan or aScan connected event.

See also

- addAccessoryDidConnectNotification: handler
- addAccessoryDidDisconnectNotification: handler
- removeAccessoryDidDisconnectNotification: handler

NotificationHandler

Parameters

handler	The object implement NotificationHandler protocol.

6.7.2.13 - (void) removeAccessoryDidDisconnectNotification: (NSObject <NotificationHandler> *) *handler*

Remove handler not to handle iPDT380, iScan or aScan disconnected event. After you remove handler, you will not trigger with iPDT380 or iScan disconnected event.

See also

- addAccessoryDidConnectNotification: handler
- removeAccessoryDidConnectNotification: handler
- addAccessoryDidDisconnectNotification: handler

NotificationHandler

Parameters

handler	The object implement NotificationHandler protocol.

6.7.2.14 - (void) removeReceiveCommandHandler: (NSObject <ReceiveCommandHandler> *) *handler*

Remove handler which handle receive command layer object event After you remove handler, you will not trigger with iPDT380, iScan or aScan receive command event.

See also

- addReceiveCommandHandler: handler
ReceiveCommandHandler

Parameters

handler	The object implement ReceiveCommandHandler protocol.

6.7.2.15 - (void) scan

Tigger scanner accessory to scan barcode

6.7.2.16 - (void) setup

Set the framework initalize and setup. Before you start using other function, you must call this method once.

6.7.2.17 + (id) sharedInstance

Get MLConnection instance. You can't alloc or new MLConnection instance. You can used this method to get a MLConnection instance.

Return values

*MLConnection**	MLConnection instance.

6.7.2.18 - (void) vibraMotorStrength: (NSUInteger) *i*

set scanner accessory vibra motor strength

Parameters

i	0=off, 1=small, 2=medium, 3=big

6.8 <NotificationHandler> Protocol Reference

iPDT380, iScan or aScan connected or disconnected event handler must implement this protocol.

Public Member Functions

- (void) - connectNotify
- (void) - disconnectNotify

6.8.1 Detailed Description

iPDT380, iScan or aScan connected or disconnected event handler must implement this protocol.

6.8.2 Member Function Documentation

6.8.2.1 - (void) connectNotify

When hardware connected, it will callback this method (if you addAccessoryDidConnectNotification to MLConnection).

See also

MLConnection addAccessoryDidConnectNotification

6.8.2.2 - (void) disconnectNotify

When hardware disconnected, it will callback this method (if you addAccessoryDidDisconnectNotification to MLConnection).

See also

MLConnection addAccessoryDidDisconnectNotification

6.9 <ReceiveCommandHandler> Protocol Reference

Recevie command handler must implement this protocol.

Public Member Functions

- (BOOL) - isHandler:
- (void) - handleRequest:
- (void) - handleInformationUpdate

6.9.1 Detailed Description

Recevie command handler must implement this protocol.

6.9.2 Member Function Documentation

6.9.2.1 - (void) handleInformationUpdate

After call MLScanner system trigger command, for example like batteryRemain. You will got Barcode Framework callback notification in this method.

6.9.2.2 - (void) handleRequest: (NSObject< ReceiveCommandProtocol > *) *command*

After Barcode framework recevie command and isHandler: returns TRUE, it will call this method and let handler take care this command.

Parameters

command	The object implement ReceiveCommandProtocol protocol which can handle receive command.

See also

- isHandler:

6.9.2.3 - (BOOL) isHandler: (NSObject< ReceiveCommandProtocol >*) *command*

After Barcode framework recevie command, it will callback this method to make sure that handler will handle it or not. If return TRUE, framework will call method: handleRequest:.

Parameters

command	The object implement ReceiveCommandProtocol protocol which can return will handle receive command or not.

Return values

TRUE	The handler will handle this command.
FALSE	The handler will not handle this command.

See also

- handleRequest:

6.10 ScanShot Class Reference

A command to trigger iPDT380, iScan or aScan to scan barcode.

Inherits ExecuteCommand.

6.10.1 Detailed Description

A command to trigger iPDT380, iScan or aScan to scan barcode.

