Using uFCoder library in Xcode

1.4

Digital Logic Ltd.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Table of contents

iOS development 3
Download iOS library 3
Linking static uFCoder library for iOS app development 3
iOS uFCoder library usage 4
Necessary information for projects “info.plist” file 4
BLE usage 4
NFC usage 4
macOS development 7
Download macOS Universal libraries 7
Linking macOS uFCoder library 7
uFCoder - import C functions to Swift 9
C types 9
Pointers 10
Examples 10
GetCardIdEx() 10
C declaration 10
Swift declaration 10
Swift code 11
BlockWrite() 11
C declaration 11
Swift declaration 11
Swift code 11
BlockRead() 12
C declaration 12
Swift declaration 12
Swift code 12
Revision history 13

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

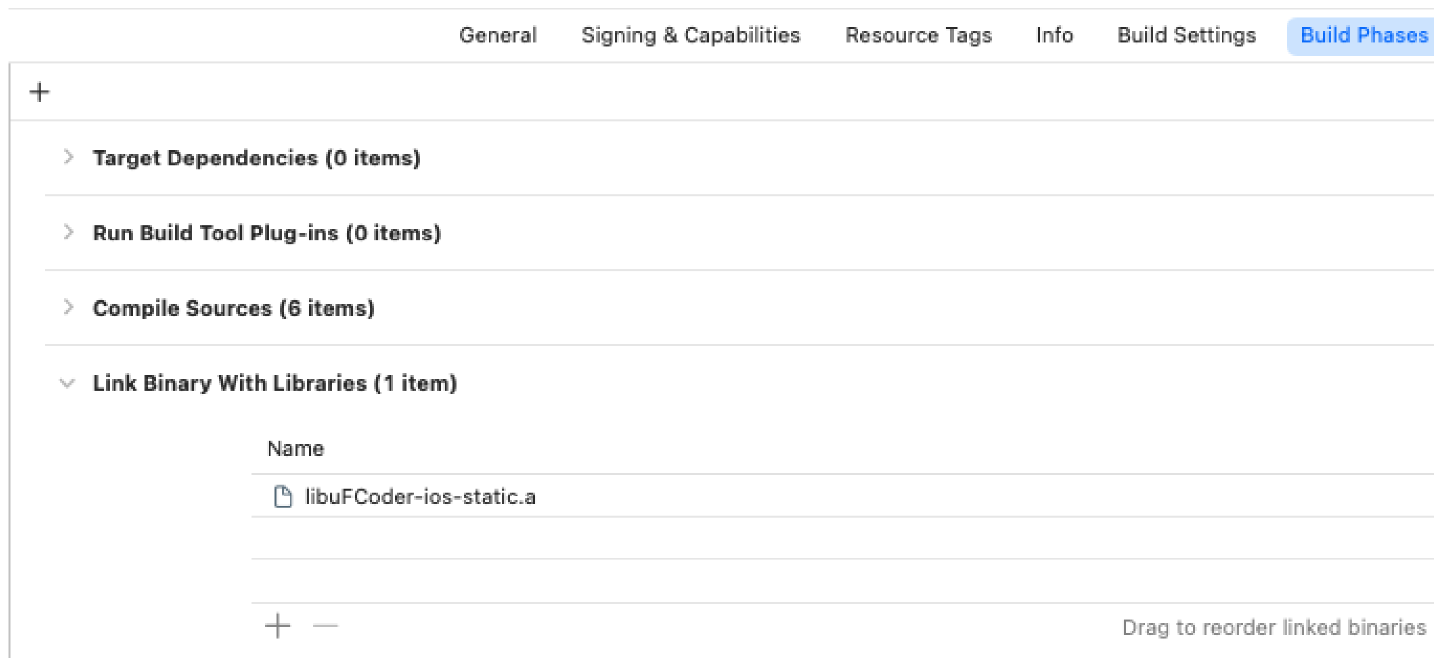
iOS development

Download iOS library

uFCoder libraries can be found here:
https://www.d-logic.com/code/nfc-rfid-reader-sdk/ufr-lib/tree/master/ios

Linking static uFCoder library for iOS app development

1. Open your project settings in XCode, and go to “**Build Phases**”
2. Under “**Link Binary With Libraries**” click on “**+**” to add the “**libuFCoder-ios-static.a**” file in your project. (If you are using our ufr-lib repository, path of this file should resemble ”/ufr-lib/ios/libuFCoder-ios-static.a”)



3. Depending on whether using **Swift** or **Objective-C**, you will need to include “uFCoder.h” header file in your project
   - **3.1. Swift**
     - 3.1.1. Add a bridging header to Xcode project (**File** > New > File > **Header file)**
     - 3.1.2. For example name the new file “YourProjectName-Bridging-Header.h”
     - 3.1.3. Create the file and add “#import “<path_to_uFCoder.h_file>”
       (e.g “#include “ufr-lib/include/uFCoder.h”)

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

3.1.4. Navigate to your project **build settings** and find the “**Swift Compiler - General**” section

3.1.5. Add path of our newly created bridging header file in the value field of “**Objective-C Bridging Header**” field

3.2. Objective-C

No bridging header is necessary for Objective-C.
Simply add “#import “<path_to_uFCoder.h_file>” after you link your project with uFCoder library, as explained in step 2 and you’re finished with setting up usage of uFCoder library in your application.

iOS uFCoder library usage

uFCoder library for iOS currently only supports our uFR Nano Online NFC reader. More specifically, it supports UDP, TCP and BLE communication with this device. There is also support for using the NFC antenna on your iOS device using our library for sending and receiving data, however it only supports APDU commands using the ApduPlainTransceive() function (demo code will be provided in the following text). When using BLE or phone NFC, your project's “info.plist” file must be updated accordingly, with necessary information about permissions and capabilities of your app.

Necessary information for projects “info.plist” file

BLE usage

If you plan on using BLE, you need to add the following keys with a message in your projects “info.plist” file:

- “privacy - bluetooth always usage description”
- “privacy - bluetooth peripheral usage description”

Refer to the official documentation for these keys:
https://developer.apple.com/documentation/corebluetooth

To connect to uFR Nano Online when the reader is in BLE mode, use **ReaderOpenEx()** function with following parameters:

- Reader type: 0
- Port name: ONxxxxxx - serial number of the reader (e.g ON123456_BLE)
- Port interface: ‘L’ or 76 (decimal)
- Additional argument: null (not a necessary parameter)

ReaderOpenEx() should return **UFR_OK** on success and readers RGB colors will be steady light-blue.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

NFC usage

For now, you can use NFC only to send APDU commands via the device's NFC antenna.
You will need to add following in the "info.plist" file:

- "com.apple.developer.nfc.readersession.formats" - NFC data formats an app can read. This entitlement requires you to add "Near Field Communication Tag Reading" capability. This entitlement should be an array of strings in the info.plist file, add the "TAG" string as a value for this entitlement.
- "ISO7816 application identifiers for NFC Tag Reader Session" - list of application identifiers that the app supports. This entitlement is an array, and should contain following values:
  - A0000002471001
  - D2760000850101
  - 00000000000000
- "Privacy - NFC Scan Usage Description" - message that tells the user why the app is requesting access to the device's NFC hardware.

Use **ReaderOpenEx()** function with the following parameters:

- Port name: 5
- Port name: ""
- Port Interface: 0
- Additional argument: null or 0

To send an APDU command, connecting to the tag is necessary, to achieve this - SetISO14443_4_Mode() function is used.

On successful connection, UFR_OK is returned, and the tag is ready to receive APDU command(s).

To send an APDU command - ApduPlainTransceive() function is necessary.
Once you're done with sending the APDU commands, call **s_block_deselect()** function.

Whole process should look like this: (Swift code bellow):

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

```
72
73      @IBAction func onSendApdu(_ sender: Any) {
74          var status: UFR_STATUS = ReaderOpenEx(5, "", 0, nil)
75          print("ReaderOpenEx_status: \(status)")
76
77          status = SetISO14443_4_Mode()
78          print("SetISO14443_4_Mode: \(status)")
79          if (status != UFR_OK)
80          {
81              print("SetISO14443_4_Mode failed")
82              return
83          }
84              let str_apdu = "00A4040C07A0000002471001"
85              let capdu = str_apdu.hexa
86              let clen: UInt32 = UInt32(capdu.count)
87              var rapdu = [UInt8](repeating: 0x00, count: 10)
88              var rlen: UInt32 = 1
89
90              status = APDUPlainTransceive(capdu, clen, &rapdu, &rlen)
91              print("APDUPlainTransceive: \(status)")
92
93              s_block_deselect(100)
94      }
95  }
96
97  extension Sequence where Element == UInt8 {
98      var data: Data { .init(self) }
99      var hexa: String { map { .init(format: "%02x", $0) }.joined() }
100 }
101
102
```


Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · **Tel:** +38112541022· **VAT:** 111385444 **Reg.** 21473642
e-mail: office@d-logic.com · www.d-logic.com

macOS development

As of version 5.0.84 - users have access to uFCoder **Universal libraries** for macOS that can be used natively on both Apple silicon and Intel-based Mac computers.
Supported architectures: x86_64 and arm64.

Download macOS Universal libraries

Both dynamic (.dylib) and static (.a) libraries are provided in our SDK repository.
They can be found here:
https://www.d-logic.com/code/nfc-rfid-reader-sdk/ufr-lib/tree/master/macos

Subfolders are:

- x86_64: dynamic library for Intel x86_64 only
- static-x86_64: static library for Intel x86_64 only
- **universal**: dynamic universal library for both x86_64 and arm64
- **static-universal**: static universal library for both x86_64 and arm64

Linking macOS uFCoder library

1. Open your project settings in XCode, and go to "**Build Phases**"
2. Under **Link Binary With Libraries**" click on "**+**" to add the "**libuFCoder-macos.dylib**" file in your project. (If you are using our ufr-lib repository, path of this file should be something like "/ufr-lib/ios/uFCoder-ios-static.a")
3. Depending on whether using **Swift** or **Objective-C**, you will need to include "uFCoder.h" header file in your project

3.1. Swift

3.1.1.1. Add a bridging header to Xcode project (**File** > New > File > **Header file)**

3.1.1.2. For example name the new file "YourProjectName-Bridging-Header.h"

3.1.1.3. Create the file and add "#import "<path_to_uFCoder.h_file>"
(e.g "#include "ufr-lib/include/uFCoder.h")

3.1.1.4. Navigate to your project build settings and find the "**Swift Compiler - General**" section

3.1.1.5. Add path of our newly created bridging header file in the value field of "**Objective-C Bridging Header**" field

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

3.2. Objective-C

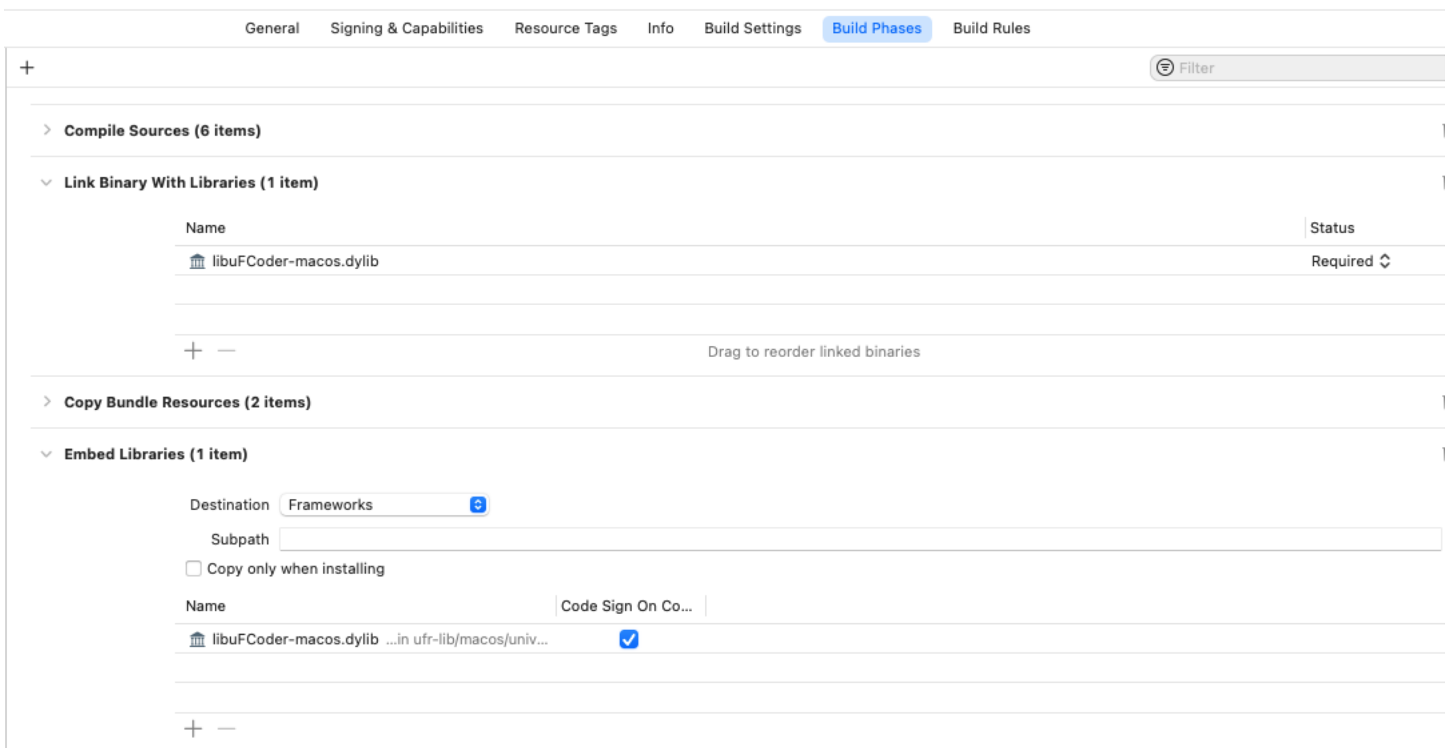
3.2.1.1. No bridging header is necessary for Objective-C.
Simply add "#import "<path_to_uFCoder.h_file>" after you link your project with uFCoder library, as explained in step 2 and you're finished with setting up usage of uFCoder library in your application.

Important: When linking the **dynamic** library make sure the following settings are valid:

1. **Build Phases -> Link Binary with Libraries**: Add the **libuFCoder-macos.dylib**
2. Go to tab **General** and make sure the uFCoder library is embed in the **"Framewok, Libraries and Embedded Content"**

Frameworks, Libraries, and Embedded Content

After the library is set to be embed, **Build phases** will add **Embed Libraries** step automatically, with the default of copying the library to the apps **Frameworks** directory



3. Set the **Runpath Search Paths** as: **@executable_path/../Frameworks**
4. Finally, if you plan on using **App Sandbox** make sure your **entitlements** file contains necessary permissions:

Key	Type	Value
Entitlements File	Dictionary	(5 items)
App Sandbox	Boolean	YES
com.apple.security.device.usb	Boolean	YES
Outgoing Network Connections	Boolean	YES
com.apple.security.device.serial	Boolean	YES
com.apple.security.network.server	Boolean	YES

uFCoder - import C functions to Swift

Due to uFCoder library being native C library, parameters will need to be configured properly and adjusted to be compatible with C types when importing and using methods from *uFCoder.h*.

C types

Frequently used C types in *uFCoder.h* are:

C	Swift Type
bool	Bool
char, unsigned char (int8_t, uint8_t)	Int8, UInt8
short, unsigned short (int16_t, uint16_t)	Int16, UInt16
int, unsigned int (int32_t, uint32_t)	Int32, UInt32
long long, unsigned long long (int64_t, uint64_t)	Int64, UInt64

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Pointers

C Type	Swift Type
char*, unsigned char* (int8_t*, uint8_t*)	UnsafeMutablePointer<Int8>! UnsafeMutablePointer<UInt8>!
short, unsigned short (int16_t, uint16_t)	UnsafeMutablePointer<Int16>! UnsafeMutablePointer<UInt16>!
int, unsigned int (int32_t, uint32_t)	UnsafeMutablePointer<Int32>! UnsafeMutablePointer<UInt64>!
long long, unsigned long long (int64_t, uint64_t)	UnsafeMutablePointer<Int64>! UnsafeMutablePointer<UInt64>!
void*	UnsafeMutableRawPointer!

With the *void* (UnsafeMutableRawPointer)* as an exception, most often these pointers refer to a single variable (**VAR** parameters in uFCoder.h) used to return some value, or an array being passed/returned (**IN**/**OUT** parameters in uFCoder.h)

Refer to **uFR Series NFC Reader API** for details about the parameters.
Git repository for documentation: https://www.d-logic.com/code/nfc-rfid-reader-sdk/ufr-doc.git

Examples

GetCardIdEx()

C declaration

```
GetCardIdEx(VAR uint8_t* lpucSak, OUT uint8_t *aucUid, VAR uint8_t *lpucUidSize);
```

Swift declaration

```
GetCardIdEx(lpucSak: UnsafeMutablePointer<UInt8>!, aucUid:
UnsafeMutablePointer<UInt8>!, lpucUidSize: UnsafeMutablePointer<UInt8>!)
```


Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Swift code

```
var status: UFR_STATUS = UFR_COMMUNICATION_BREAK

var lpucSak: UInt8 = 0
var aucUid: [UInt8] = [UInt8](repeating: 0, count: 11)
var lpucUidSize: UInt8 = 0

status = GetCardIdEx(&lpucSak, &aucUid, &lpucUidSize)
…
```

BlockWrite()

C declaration

```
BlockWrite(IN const uint8_t *data, uint8_t block_address, uint8_t auth_mode,
uint8_t key_index);
```

Swift declaration

BlockWrite(data: UnsafePointer<UInt8>!, block_address: UInt8, auht_mode: UInt8, key_index: UInt8)

Swift code

```
var status: UFR_STATUS = UFR_COMMUNICATION_BREAK

var data: [UInt8] = [UInt8](repeating: 0, count: <data_len>)
var block_address: UInt8 = 0
var auth_mode: UInt8 = 0x60
var key_index: UInt8 = 0

status = BlockWrite(data, block_address, auth_mode, key_index)
…
```


Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

BlockRead()

C declaration

```
BlockRead(OUT uint8_t *data, uint8_t block_address, uint8_t auth_mode, uint8_t
key_index);
```

Swift declaration

```
BlockRead(data: UnsafeMutablePointer<UInt8>!, block_address: UInt8, auht_mode:
UInt8, key_index: UInt8)
```

Swift code

```
var status: UFR_STATUS = UFR_COMMUNICATION_BREAK

var data: [UInt8] = [UInt8](repeating: 0, count: <data_len>)
var block_address: UInt8 = 0
var auth_mode: UInt8 = 0x60
var key_index: UInt8 = 0

status = BlockRead(&data, block_address, auth_mode, key_index)
…
```


Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Revision history

Date	Version	Comment
2023-02-27	1.4	Linking macOS uFCoder library section updated. uFCoder - import C functions to Swift section added. Examples added.
2023-02-27	1.3	Document renamed. macOS development section added.
2023-01-24	1.2	Download section added.
2021-10-29	1.1	BLE usage descriptions updated
2019-04-09	1.0	Base document

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com