AqKanji2Koe iOS マニュアル

株式会社 アクエスト
www.a-quest.com

概要

本文書は、言語処理ライブラリ AqKanji2Koe iOS をアプリケーションに組み込んで使用するためのプログラミングの方法、注意点を示したものです。

AqKanji2Koe は、漢字かな混じり文のテキスト情報を AquesTalk 用のアクセント付きの音声記号列に変換するライブラリです。
このライブラリと音声合成ライブラリ AquesTalk を使えば、動的に変化する様々なテキストから、リアルタイムに音声メッセージを生成できるようになります。

特長
・簡単に組み込み可能
　- テキスト文字列を入力すると音声記号列を返す、シンプルな API
・高速な変換処理
　- 約 20 万字/秒の高速な変換処理
・登録単語の追加が可能
　- ユーザ辞書ライブリを用いて、アプリで単語の追加や、編集、削除が可能
・高精度な読み・アクセント付与
　- 最大 50 万語の単語辞書とアクセントルールにより、正確な読みとアクセントを生成

本ライブラリを使ったプログラミングに先立って動作を確認される場合は、弊社サイトのオンラインデモや、言語処理エンジンのページからデモアプリ AqK2kDemo をダウンロードしてお試しください。漢字を含んだ文を音声記号列に変換や、ユーザ辞書を編集する機能を確認できます。

オンラインデモ	https://www.a-quest.com/demo/
デモアプリ ダウンロード	https://www.a-quest.com/products/aqkanji2koe.html

本ライブラリを使用するには開発ライセンスキーの設定が必要です。このライセンスキーを設定しない場合は評価版として動作し、以下の制限があります。

評価版の制限	**「ナ行、マ行」は、すべて「ヌ」と出力されます**

本ライブラリをアプリケーションに組み込んで使用する際には**使用ライセンス**、配布には**頒布ライセンス**が必要です。ライセンスの種類や購入方法は、弊社サイトのライセンスのページを参照してください。

仕様

ライブラリ形式	iOS 用 static ライブラリ arm64,armv7s,armv7,x86_64,i386,bitcode
対応 OS	iOS 6 以上
入力データ形式	漢字かな混じり文テキスト(UTF8)
出力データ形式	かな表記音声記号列(UTF8)
関数 I/F	C 関数呼び出し
マルチスレッド	対応
ライブラリサイズ	約 180KByte(リンク後 arm64)
辞書サイズ	標準:約7MB(約36万語)、スモール:約5MB(約25万語) ラージ:約13MB(約50万語)
処理速度	約 20 万文字/秒 (Windows XP, AthlonX2 2.7GHz 環境で計測)
ビルド環境	LLVM 9.0
依存ライブラリ	libc++

ビルド・実行

ライブラリ使用方法・注意点

AqKanji2Koe Linux のライブラリはスタティックライブラリです。アプリのリンク時に必要で、実行時には不要です。

アプリケーションに組み込む際のポイントは、以下の 3 点です。

- ヘッダファイル AqKanji2Koe.h とライブラリ libAqKanji2Koe.a をプロジェクトに追加
- 辞書ファイル一式をアプリバンドル内に含める
- ライブラリ libc++を"リンクするライブラリ"に追加

辞書ファイル(aqdic.bin,aq_user.dic,CREDITS)はアプリバンドル(<アプリ名>.app)内に含まれるよう、Target > Build Phases > Copy Bundle Resources に追加します。バンドル直下のファイルはアクセスできませんので、aq_dic などのフォルダごとコピー・配置します。

libc++のプロジェクトに追加を忘れると、リンク時に大量のエラーが起きます。
Target > Build Phases > Link Binary With Libraries に libc++.tbd が含まれていることを確認してください。

ユーザ辞書ライブラリ libAqUsrDic

ユーザ辞書に単語の追加や削除、編集を行う場合には、別途、ユーザ辞書ライブラリ libAqUsrDic を用います。アプリに組み込む際は、libAqKanji2Koe と同様の方法で、AqUsrDic.h, libAqUsrDic.a を使用します。
ユーザ辞書ファイル(aq_user.dic)はシステム辞書ファイル(aqdic.bin)に依存するため、必ず同じフォルダに配置する必要があります。しかし、iOS の制約でアプリバンドル内のファイルは書き換えできないため、バンドル内のユーザ辞書ファイルは更新できません。そこで、ユーザ辞書ライブラリを使用する際には、aq_user.dic と aqdic.bin を Library フォルダなどのアプリバンドルの外の読み書き可能なフォルダにコピーして、そのフォルダ上で処理を行います(サンプルアプリではこの方法を使っています）。

アーキテクチャと Universal Binary

libAqKanji2Koe は、Simulator(シミュレータ）用と Device(実機)用の２つの異なるアーキテクチャのライブラリがひとつにパックされている Universal Binary になっています。Xcode がビルド時にターゲットにあわせて適切な方を選択するので、開発者がターゲットに応じて切り替える必要は有りません。

なお、本ライブラリには、Device 用として、armv7(iPhone4S 以前）、armv7s(iPhone5)、arm64(iPhone5s 以降)の３つ、Simulator 用として i386 と x86_64 の２つの計５つのアーキテクチャのライブラリが含まれています。

さらに、bitcode にも対応し、iOS9 からのアプリ最適化の仕組み AppThinning 用の bitcode に対応したアプリも作成できます。

関数 API

libAqKanji2Koe.a

AqKanji2Koe_Create　　AqKanji2Koe.h

説明	言語処理モジュールのインスタンス生成と内部データの初期化 生成したインスタンスは、使用後、AqKanji2Koe_Release で解放してください。
構文	void * **AqKanji2Koe_Create** (const char **pathDic*, int **pErr*)
引数	
pathDic	辞書のディレクトリを指定。通常、"<app dir>/aq_dic"。指定した内容は内部で保存される。 文字列最後の、/　の付与は任意
pErr	エラー時にはエラーコードが入る 正常終了時は不定値
戻り値	インスタンスハンドル エラーの時は0が返る。このとき pErr にエラーコードが設定される。

AqKanji2Koe_Create_Ptr

AqKanji2Koe.h

説明 言語処理モジュールのインスタンス生成と内部データの初期化
生成したインスタンスは、使用後、AqKanji2Koe_Release で解放してください。

構文 void * **AqKanji2Koe_Create_Ptr** (const void **pSysDic*, const void **pUserDic*, int **pErr*)

引数

pSysDic システム辞書(通常 aqdic.bin) をメモリ上に読み込んだ先頭アドレスを指定

pUserDic ユーザ辞書(通常 aq_user.dic) をメモリ上に読み込んだ先頭アドレスを指定
ユーザ辞書を使用しない場合は NULL を指定する

pErr エラー時にはエラーコードが入る 正常終了時は不定値

戻り値 インスタンスハンドル エラーの時は0が返る。このとき pErr にエラーコードが設定される。

AqKanji2Koe_Release

AqKanji2Koe.h

説明 言語処理モジュールのインスタンスを開放

構文 void **AqKanji2Koe_Release** (void * *hAqKanji2Koe*)

引数

hAqKanji2Koe AqKanji2Koe_Create() / AqKanji2Koe_Create_Ptr()で返されたハンドルを指定

戻り値 なし

AqKanji2Koe_Convert

AqKanji2Koe.h

説明 漢字かな混じりのテキストを音声記号列に変換(UTF8 版)

構文 int **AqKanji2Koe_Convert** (void * *hAqKanji2Koe*, const char **kanji*, char **koe*, int *nBufKoe*)

引数

hAqKanji2Koe AqKanji2Koe_Create() / AqKanji2Koe_Create_Ptr()で返されたハンドルを指定

kanji 入力漢字かな混じり文テキスト文字列（UTF8, NULL 終端)

koe 出力バッファ。音声記号列文字列が返る（UTF8, NULL 終端)

nBufKoe koe バッファのサイズ[byte] 256 以上を指定。バッファサイズ以上の音声記号列は切り捨てられますので入力テキストの数倍のサイズを指定することを推薦。

戻り値　0:正常終了　それ以外:エラーコード

AqKanji2Koe_ConvertW　AqKanji2Koe.h

説明　漢字かな混じりのテキストを音声記号列に変換(UTF32 版)

構文　int **AqKanji2Koe_Convert** (void * *hAqKanji2Koe*, const wchar_t **kanji*, wchar_t **koe*, int *nBufKoe*)

引数

hAqKanji2Koe　AqKanji2Koe_Create() / AqKanji2Koe_Create_Ptr()で返されたハンドルを指定

kanji　入力漢字かな混じり文テキスト文字列(Unicode(UTF32), NULL 終端)

koe　出力バッファ。音声記号列文字列が返る(Unicode(UTF32), NULL 終端)

nBufKoe　koe バッファのサイズ[byte] 256 以上を指定。バッファサイズ以上の音声記号列は切り捨てられますので入力テキストの数倍のサイズを指定することを推薦。

戻り値　0:正常終了　それ以外:エラーコード

AqKanji2Koe_SetDevKey　AqKanji2Koe.h

説明　開発ライセンスキーを設定。アプリ起動後、他の関数を呼び出す前に呼び出すことで、以降、製品版とし動作し、評価版の制限がなくなる。

構文　int **AqKanji2Koe_SetDevKey**(const char **key*)

引数

key　開発ライセンスキー文字列(半角英数)

戻り値　ライセンスキーが正しければ 0、正しくなければ 1 が返る。
不正なキーでも 0 を返す場合がある。このとき制限は解除されない。

libAqUsrDic.a（ユーザ辞書ライブラリ）

AqUsrDic ライブラリは、個々の単語の登録を行うインターフェースはありません。アプリケーションでユーザ辞書を変更するときは、CSV 形式の単語リストファイルを介して行います。

AqUsrDic_Import　AqUsrDic.h

説明 CSV 形式の単語リストからユーザ辞書(aq_usr.dic)を生成
生成する aq_usr.dic と同じディレクトリに、システム辞書（aqdic.bin）が必要。
aq_usr.dic がすでにある場合は上書きする(Append しない)。
【注意】ユーザ辞書は生成時のシステム辞書に依存するため、異なる(サイズの)システム辞書とは使用できない。

構文 int **AqUsrDic_Import**(const char * *pathUserDic*, const char * *pathDicCsv*)

引数

pathUserDic 出力するユーザ辞書(aq_user.dic)ファイルを指定

pathDicCsv CSV 単語リストファイルファイルを指定(入力データ)

戻り値 0:正常終了 それ以外:エラーコード(詳細は AqUsrDic_GetLastError()で取得)

AqUsrDic_Export

AqUsrDic.h

説明 ユーザ辞書(aq_usr.dic)から CSV 形式の単語リストを生成
aq_usr.dic と同じディレクトリに、システム辞書（aqdic.bin）が必要。

構文 int **AqUsrDic_Export**(const char * *pathUserDic*, const char * *pathDicCsv*)

引数

pathUserDic ユーザ辞書(aq_user.dic)ファイルを指定

pathDicCsv 出力する CSV 単語リストファイルファイルを指定

戻り値 0:正常終了 それ以外:エラーコード(詳細は AqUsrDic_GetLastError()で取得)

AqUsrDic_Check

AqUsrDic.h

説明 単語表記、読みの書式や品詞コードが正しいかチェックする

構文 int **AqUsrDic_Check**(const char * *surface*, const char * *yomi*, int *posCode*)

引数

surface 単語の表記の文字列を指定(UTF8)

yomi 単語の読み(アクセント付き音声記号列)の文字列を指定(UTF8)

posCode 単語の品詞コード(下記参照)を指定

戻り値 0:チェック OK それ以外:NG(詳細は AqUsrDic_GetLastError()で取得)

AqUsrDic_GetLastError　　AqUsrDic.h

説明	最後のエラーの詳細メッセージを返す
構文	const char * **AqUsrDic_GetLastError**()
引数	なし
戻り値	エラーメッセージ(UTF8, NULL 終端)

エラーコード表

libAqKanji2Koe ライブラリの関数が返すエラーコードの内容は、次の通りです。
libAqUsrDic ライブラリに関しては、AqUsrDic_GetLastError()関数で内容を取得してください。

値	内容
100	その他のエラー
101	関数呼び出し時の引数が NULL になっている。
104	初期化されていない(初期化ルーチンが呼ばれていない)
105	入力テキストが長すぎる
106	システム辞書データが指定されていない
107	変換できない文字コードが含まれている
200 番台	システム辞書(aqdic.bin)が不正
300 番台	ユーザ辞書(aq_user.dic)が不正

辞書サイズ

この SDK には、サイズの異なる 3 種類のシステム辞書が含まれています。
利用の目的に応じて選択してお使いください。

辞書名	フォルダ	サイズ	見出し語数
標準辞書	aq_dic	約7MB	約 36 万語
ラージ辞書	aq_dic_large	約 13MB	約 50 万語
スモール辞書	aq_dic_small	約 5MB	約 25 万語

CSV 単語リストファイル

ユーザ辞書の編集で使用する CSV 単語リストファイルの例を下に示します。
1 列目：単語の表記を記述します。文字コードは UTF8 です。すべて全角で記述します。空白は指定できません。
2 列目：単語の読みを音声記号列で指定します(UTF8)。アクセントも ' 記号で指定できます。詳細は音声記号列仕様書の読み記号とアクセント記号の項を参照ください。区切記号は指定できません。
3 列目：単語の品詞コード(下記参照)を半角数値で指定します。

CSV 単語リストファイルの例

```
大江戸線,オーエドセン,5
ＧＤＧＤ,グ'ダグダ,1
天王洲,テンノーズ,7
Ｊアラート,ジェイアラ'ート,5
就活,シューカツ,1
魔理沙,マ'リサ,4
どや顔,ドヤガオ,1
アラフォー,アラフォー,0
```

品詞コード一覧

0	名詞
1	名詞(サ変)
2	人名
3	人名(姓)
4	人名(名)
5	固有名詞
6	固有名詞(組織)
7	固有名詞(地域)
8	固有名詞(国)
9	代名詞
10	代名詞(縮約)
11	名詞(副詞可能)
12	名詞(接続詞的)
13	名詞(形容動詞語幹)
14	名詞(ナイ形容詞語幹)
15	形容詞
16	副詞
17	副詞(助詞類接続)
18	接頭詞(名詞接続)
19	接頭詞(動詞接続)
20	接頭詞(数接続)
21	接頭詞(形容詞接続)
22	接続詞
23	連体詞
24	記号
25	記号(アルファベット)
26	感動詞
27	間投詞

サンプルアプリ

本 SDK にはサンプルアプリ HelloKanji2Koe という xcode プロジェクトが含まれています。このアプリは、任意の日本語テキストを音声記号に変換、ユーザ辞書内容を CSV 出力、逆に CSV ファイルユーザ辞書を構築する機能を持っています。

なお、サンプルアプリのソースコード(ライブラリやライブラリのヘッダファイルを除く)は、ご自由に改変してご使用いただけます。

開発環境は Xcode 9.1 で確認しています。基本的に Menu > Run だけで実行できます。

上部のテキストボックスに漢字を含んだ任意のテキストを入力し、[Convert]の押下で、下部のテキストボックスに変換した音声記号列が表示されます。

また、ユーザ辞書ライブラリ AqUsrDic も使用しています。[Export]の押下で、ユーザ辞書の内容が CSV 形式でファイル出力されます。このファイルはアプリのサンドボックスの tmp フォルダ内に aq_user.csv として生成されます。ユーザ辞書内容の表示や編集機能はありませんので、実際のアプリでは別途実装してください。また、[Import]の押下で、tmp フォルダ内の aq_user.csv からユーザ辞書ファイル Library フォルダ内に aq_user.dic が再構築されます。

アプリケーションへの実装方法

AqKanji2Koe ライブラリをアプリケーションに組み込む手順を以下に示します。AqUsrDic ユーザ辞書ライブラリを組み込む際は、AqKanji2Koe と同じ方法で行います。

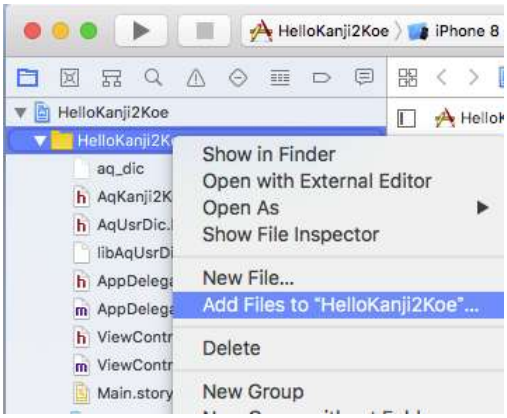
ヘッダ、ライブラリの追加

ヘッダファイル AqKanji2Koe.h をインクルードし、 ビルド時にスタティックライブラリ libAqKanji2Koe.a をリンクするようにします。

AqKanji2Koe.h は、アプリケーションのソースファイルと同じディレクトリにコピーして、ソース内で次のようにインポートします。

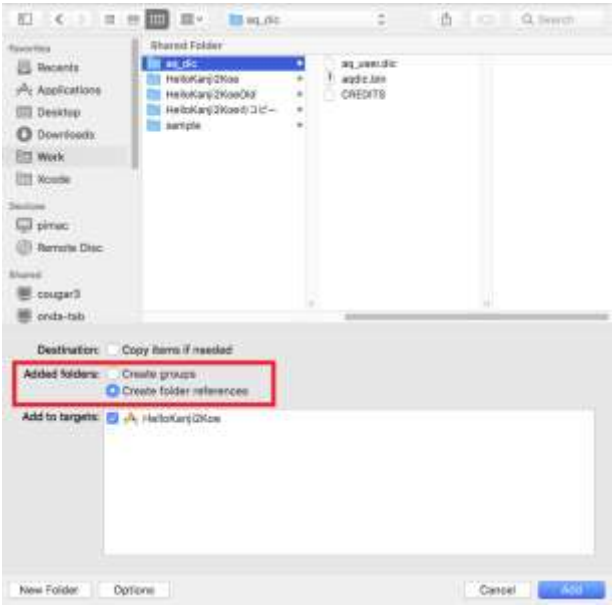
```
#import "AqKanji2Koe.h"
```

libAqKanji2Koe.a も、ソースファイルと同じディレクトリにコピーしてから、プロジェクトの右クリックメニューの Add Files to"…"でプロジェクトに追加します。



辞書ファイルの追加

辞書ファイルが実行時に必要なため、ビルド時にこれらがアプリバンドル(.app)内にコピーされるように設定します。



SDK にはサイズの異なる 3 つの辞書ファイルが含まれていますので、いずれかを選択します。以下の例では標準辞書(aq_dic フォルダ）を使います。

プロジェクトの右クリックメニューの Add files to “xxx”… で aq_dic フォルダを選択します。このとき、[Options]の[Added folders]が[Create folder references]が選択されていることを確認してください。もし、[Create groups]が選択されていると、aq_dic が黄色のアイコンとなり、実行時に辞書ファイルを読めません。

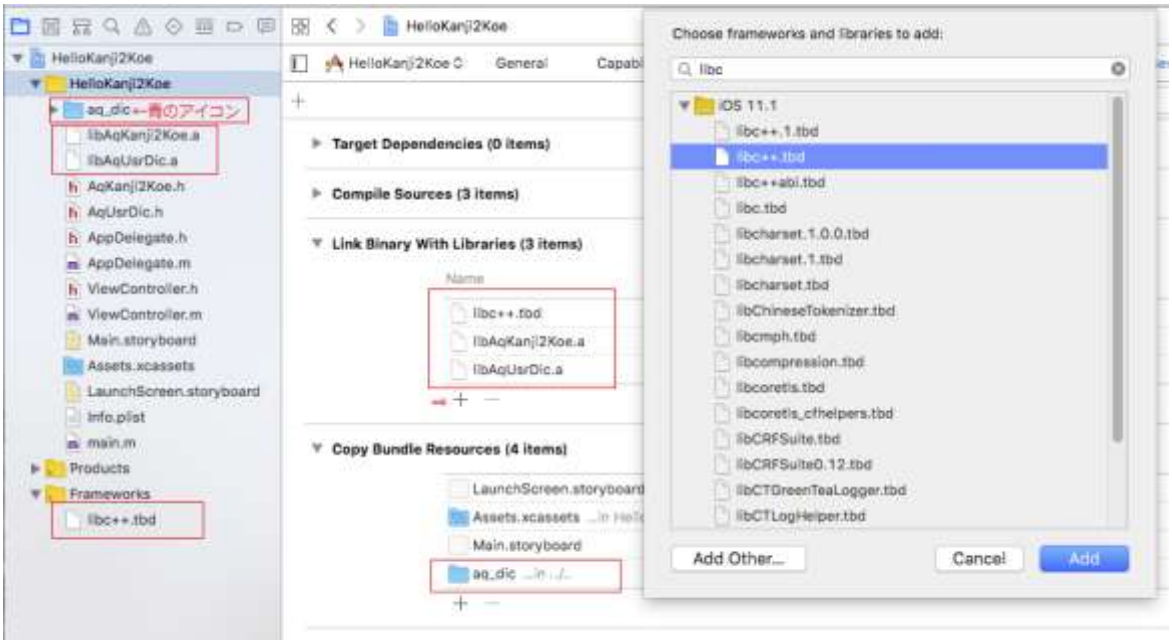
これで、プロジェクトナビゲーター上に青色のフォルダアイコンの aq_dic が表示されます。また、ビルド時にアプリバンドル内に辞書ファイルが aq_dic フォルダごとコピーされます。

外部ライブラリの追加

AqKanji2Koe/AqUsrDic は、C++標準ライブラリの libc++ライブラリを使用します。これを指定し忘れると"関数が見つからない"という大量のリンクエラーがビルド時に発生します。

アプリのプロジェクトに libc++を追加するには、下図のように、Target の設定画面　General＞Build Phases >Link Binary With Libraries で[＋]部分を押下します。

なお、アプリ側で.mm などを使っていて、すでに C++のコンパイラを使用するようになっている場合は、libc++の追加は不要です。



最終的に、次のことを確認します。

1. Project Navigator(上図左側のファイルリスト)に、青のアイコンの辞書ファイルのフォルダ aq_dic とスタティックライブラリ libAqKanji2Koe.a と libc++.tbd がある。
2. Target の Build Phases（上図中央部）の Link Binary With Libraries に libc++.tbd と libAqKanji2Koke.a がある。
3. Target の Build Phases（上図中央部）の Copy Bundle Resources に辞書ファイルのフォルダ(aq_dic)がある。

ソースコード

サンプルプログラムのソースコードを例にプログラムの方法を示します。本サンプルは Objective-C で記述していますが、swift でも特に違いはありません。

ViewController.h

```
1   #import <UIKit/UIKit.h>
2
3   @interface ViewController : UIViewController
4   {
5       IBOutlet UITextField* kanji;
6       IBOutlet UITextField* koe;
7       void *m_pAqKanji2Koe;
8   }
9
10  @end
```

ViewController.h で、AqKanji2Koe インスタンス用の変数を用意します（7 行目）。

ViewController.m

```
1   - (void)viewDidLoad {
2       [super viewDidLoad];
3
4       // ユーザ辞書の書換えのため、辞書ファイルをアプリバンドルから Library フォルダにコピー
5       //  ユーザ辞書の書換え不要ならアプリバンドルのままでも OK
6       [self copyDicFiles];
7
8       // 開発ライセンスキーのセット
9       int iret = AqKanji2Koe_SetDevKey("xxx-xxx-xxx");
10      // 言語処理ライブラリのインスタンス生成
11      NSString *home = NSHomeDirectory();
12     const char *dicDir = [[home stringByAppendingString:@"/Library"] UTF8String];
13      int iErr;
14      m_pAqKanji2Koe = AqKanji2Koe_Create(dicDir, &iErr);
15      //！エラー未処理　エラーのときは NULL が戻る
16  }
17
18  - (void)dealloc {
19      // 言語処理ライブラリのインスタンス解放
```

```
20      if(m_pAqKanji2Koe) AqKanji2Koe_Release(m_pAqKanji2Koe);
21  }
```

本サンプルでは、ユーザ辞書を書き換える機能を実装するため、辞書データをアプリバンドル内からサンドボックスの Library フォルダに、システム辞書ファイル aqbin.dic とユーザ辞書ファイル aq_user.dic をコピーします。これは、別途実装した copyDicFiles メソッドで行っています（6 行目）。

開発ライセンスキーを指定します（9 行目）。引数には、ライセンス証に記載されている開発ライセンスキーを指定します。ライセンスキーが設定されていない場合は、評価版としての制限付きの動作となります。

コピーした辞書ファイルのパスを取得し（11-12 行目）、これを引数にして AqKanji2Koe_Create()を呼び出します（14 行目）。戻り値はインスタンスのアドレスです。辞書のパスが異なっていたり、辞書ファイルの内容が正しくない場合は、戻り値は NULL、iErr にエラーコードを返しますので、必要に応じてエラー処理を加えてください。

View の終了時に AqKanji2Koe のインスタンスを開放します（20 行目）

本サンプルでは、ViewController.m の viewDidLoad()内で AqKanji2Koe のインスタンスを生成し、dealloc()のタイミングでインスタンスを解放しています。このインスタンスは発声終了時まで維持する必要があります。どのタイミングで AqKanji2Koe のインスタンスを生成し解放するかは、アプリでの利用方法により変更してください。

ViewController.m

```
1   - (IBAction)convert:(id)sender
2   {
3       char cKoe[1000];
4       NSString *strKanji = [kanji text];
5
6       // NSString 文字列を C 言語文字列(Utf8)に変換
7       const char *cKanji = [strKanji UTF8String];
8
9       // 言語処理　漢字仮名まじり文を音声記号列に変換
10      AqKanji2Koe_Convert(m_pAqKanji2Koe, cKanji, cKoe, 1000);
11
12      // C 言語文字列(Utf8)を NSString に変換してテキストボックスへセット
13      koe.text = [NSString stringWithCString:cKoe encoding:NSUTF8StringEncoding];
14  }
```

メソッド convert は、[Convert]ボタンをタップしたときに呼び出されるメソッドです。ここで漢字混じりのテキストを音声記号列に変換します。

AqKanji2Koe の文字コードは UTF-8 ですので、NSString の文字列変数を、UTF-8 の C 文字列(char*)に変換します（7 行目）。

AqKanji2Koe_Convert()関数で、漢字かな混じりの文字列を音声記号列に変換します（10 行目）。変換後の音声記号列は cKoe 配列に入ります。AqKanji2Koe_Convert ()関数の最後の引数は cKoe の配列のサイズを指定します。その大きさは入力の文字の長さに応じてサイズを適宜決定してください。なお、配列サイズが音声記号列の文字列より小さい場合、適当な部分で切り捨てられてしまいます。

返される音声記号列も UTF-8 の C 文字列なので、NSString に変換後、テキストボックスにセットします（13 行目）。

ViewController.m

```
1   - (IBAction)exportUsrDic:(id)sender {
2       NSString *home = NSHomeDirectory();
3       const char *pathUsrDic = [[home stringByAppendingString:@"/Library/aq_user.dic"] UTF8String];
4       NSString *tmp = NSTemporaryDirectory();
5       const char *pathDicCsv = [[tmp stringByAppendingString:@"aq_user.csv"] UTF8String];
6
7       // ユーザ辞書 aq_user.dic の内容を、ファイル aq_user.csv に CSV 出力
8       int iret = AqUsrDic_Export(pathUsrDic, pathDicCsv);
9
10      //結果の Alert 表示
11      UIAlertController *alert;
12      if(iret==0){
13          alert = [UIAlertController alertControllerWithTitle:@"Export" message:@"CSV 出力完了"
14                                  preferredStyle:UIAlertControllerStyleAlert];
15      }
16      else {
17          alert = [UIAlertController alertControllerWithTitle:@"Export" message:@"CSV 出力失敗"
18                                  preferredStyle:UIAlertControllerStyleAlert];
19      }
20      [alert addAction:[UIAlertAction actionWithTitle:@"はい" style:UIAlertActionStyleDefault
21                                  handler:^(UIAlertAction *action) { [self otherButtonPushed];}]];
22      [self presentViewController:alert animated:YES completion:nil];
23  }
```

メソッド exportUsrDic は、[Export]ボタンをタップさしたときに呼び出されるメソッドです。ここでは、AqUsrDic ライブラリを使用して、ユーザ辞書の内容を CSV ファイルに書き出しています。

ユーザ辞書 aq_user.dic のパスと、出力する CSV ファイルのパスを取得します（2-5 行目）。CSV ファイルのパスは書き込み可能であれば任意ですが、サンドボックス内の tmp フォルダ内が良いでしょう。

AqUsrDic_Export()関数でユーザ辞書の内容を CSV ファイルに書き出しています。実際のアプリでは、この後に、書き出した CSV ファイルを読み込み、内容をリスト表示・編集する機能を実装します。
なお、単語の編集や追加の際には、AqUsrDic_Check()関数を呼び出して、内容が正しいかチェックしてください。

[Import]ボタンのタップで呼び出されるメソッド importUsrDic は、exportUsrDic とは逆に、CSV ファイルの内容でユーザ辞書 aq_user.dic を再構築しています。

AqKanji2Koe のインスタンスは AqKanji2Koe_Create 内でユーザ辞書をメモリ上に読み込んでいます。更新したユーザ辞書を利用するには、AqKanji2Koe のインスタンスを作り直して、更新したユーザ辞書を再読み込みする必要があります。

アプリ開発ガイドライン

アプリケーションの開発(評価での使用を除く)は、以下のガイドラインに従ってください。

ライセンスキー

本ライブラリの動作は、開発ライセンスキーに依存します。このキーは、各ライセンス購入時に発行されるライセンス証に記載されています。

AqKanji2Koe_SetDevKey() をアプリケーションの起動初期に一度呼び出します。引数には開発ライセンスキーを指定します。これにより製品版として動作し、評価版の制限がなくなります。

CREDITS

本ライブラリは、BSD ライセンスに基づいてライセンスされている下記のオープンソースソフトウェアを使用しています。このライセンスの表示は SDK 付属の CREDITS ファイルを参照ください。また、本ライブラリを含んだアプリを配布する場合は、CREDITS ファイルまたはその内容が含まれるようにしてください。

- MARISA: Matching Algorithm with Recursively Implemented StorAge
- NAIST Japanese Dictionary：形態素解析用辞書

文書履歴

日付	版	変更箇所	更新内容	更新者
2011/01/17	1.0	新規作成		N.Y
2013/07/03	2.0		Ver.2 用に書き換え	N.Y
2014/12/23	2.1		64bit 追加＆ビルド方法 現 xcode に合わせて修正	N.Y
2016/03/19	2.1.1		Xcode7 用に、x86_64,bitcode 記述追加	N.Y
2017/11/30	3.0		Ver.3 用に書き換え	N.Y