AqKanji2Koe-A Win 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-A Win 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-A 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。
なお、AquesTalk pico 用のローマ字の音声記号列を出力する関数も含まれています。

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
　- 約 38 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

なお、OS を持たない小型組み込み機器用に、より小型軽量の AqKanji2Koe-M もあります。こちらにはパッケージ版が無く、すべて環境に応じてビルドして提供するカスタム製品となります。

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、Windows アプリ AquesTalkPlayer 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への変換や、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/
AquesTalkPlayer	https://www.a-quest.com/products/aquestalkplayer.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	DLL(ダイナミックリンクライブラリ) 32bit / 64bit
対応 OS	各種 Windows
入力データ形式	漢字かな混じり文テキスト（UTF-8/UTF16/Shift-JIS)
出力データ形式	かな表記音声記号列（UTF-8/UTF16/Shift-JIS)
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 200KByte
辞書サイズ	約 10MB（約 38 万語）
処理速度	約 20 万文字/秒 （Windows XP, AthlonX2 2.7GHz 環境で計測）
依存ライブラリ	KERNEL32.DLL のみ（Windows OS に標準インストール済）
その他	ユーザ辞書用ライブラリ(AqUsrDic.dll) 同梱

ビルド・実行

ヘッダ、ライブラリ

プログラムのコンパイル時にはヘッダファイル(AqKanji2Koe.h)をインクルードします。必要に応じてインクルードファイルのパスを設定しておきます。リンク時には、AqKanji2Koe.lib ファイルをリンクするファイルに追加しておきます。なお、LoadLibrary()などで実行時に動的にリンクすることも可能です。

パッケージ内の lib フォルダが 32bit 版、lib64 フォルダが 64bit 版となります。

実行

アプリの実行の際には、AqKanji2Koe.dll が必要です。通常、このファイルはアプリの exe ファイルと同じディレクトリに配置します。

ユーザ辞書ライブラリ

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 AqUsrDic を用います。アプリに組み込む際は、AqKanji2Koe-A と同様の方法で、AqUsrDic.h, AqUsrDic.lib, AqUsrDic.dll を使用します。

依存ライブラリ

AqKanji2Koe.dll/AqUsrDic.dll は、コンパイルオプション/MT でビルドしています。したがって、AqKanji2Koe.dll/AqUsrDic.dll が使用する C ランタイム ライブラリ（CRT）は DLL 内に含まれています（静的リンク）。実行時に必要なライブラリは KERNEL32.dll だけで、これは Windows のシステムディレクトリに含まれています。アプリの開発において依存ライブラリを意識する必要はありません。

関数 API

AqKanji2Koe.dll

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、使用後に AqKanji2Koe_Release で解放すること。
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、/ や ￥ 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、使用後に AqKanji2Koe_Release で解放すること。 辞書データの先頭アドレスを指定するので、メモリマップドファイルでも使用可能。
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-A*)

引数

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

戻り値　なし

AqKanji2Koe_Convert_utf8　　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(UTF-8 to UTF-8)

構文　int **AqKanji2Koe_Convert_utf8** (void * *hAqKanji2Koe-A*, const char **kanji*, char **koe*, int *nBufKoe*)

引数

hAqKanji2Koe-A　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（UTF-8, BOM 無, NULL 終端)

koe　出力バッファ。音声記号列文字列が返る（UTF-8, BOM 無, NULL 終端)

nBufKoe　koe バッファのサイズ[byte] 256 以上を指定。変換後の音声記号列がこのサイズを超えるとエラーを返すので、入力テキストの2倍以上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_Convert_utf16　　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(ワイド文字（Unicode)版）

構文　int **AqKanji2Koe_Convert_utf16** (void * *hAqKanji2Koe-A*, const char16_t **kanji*, char16_t **koe*, int *nBufKoe*)

引数

hAqKanji2Koe-A　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(UTF16LE), NULL 終端)

koe 出力バッファ。音声記号列文字列が返る（Unicode(UTF16LE), NULL 終端)

nBufKoe koe バッファのサイズ[word]。バッファサイズ以上の音声記号列は切り捨てられるので入力テキストの2倍以上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_Convert_sjis　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(Shift-JIS to Shift-JIS)

構文 int **AqKanji2Koe_Convert_sjis** (void * *hAqKanji2Koe-A*, const char **kanji*, char **koe*, int *nBufKoe*)

引数

hAqKanji2Koe-A 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（ShiftJIS, NULL 終端)

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

nBufKoe koe バッファのサイズ[byte] 256 以上を指定。変換後の音声記号列がこのサイズを超えるとエラーを返すので、入力テキストの2倍以上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_ConvRoman_utf8　AqKanji2Koe.h

説明 漢字かな混じりのテキストを音声記号列(ローマ字)に変換(UTF-8 to ASCII)

構文 int **AqKanji2Koe_ConvRoman_utf8** (void * *hAqKanji2Koe-A*, const char **kanji*, char **koe*, int *nBufKoe*)

引数

hAqKanji2Koe-A 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（UTF-8, BOM 無, NULL 終端)

koe 出力バッファ。ローマ字音声記号列文字列が返る（ASCII, NULL 終端)

nBufKoe koe バッファのサイズ[byte] 256 以上を指定。変換後の音声記号列がこのサイズを超えるとエラーを返すので、入力テキストの2倍以上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_ConvRoman_utf16　　AqKanji2Koe.h

説明　漢字かな混じりのテキストを音声記号列(ローマ字)に変換(ワイド文字（Unicode)版）

構文　int **AqKanji2Koe_ConvRoman_utf16**(void * *hAqKanji2Koe-A*, const char16_t **kanji*, char16_t **koe*, int *nBufKoe*)

引数

hAqKanji2Koe-A	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(UTF16LE), NULL 終端)
koe	出力バッファ。ローマ字音声記号列文字列が返る（Unicode(UTF16LE), NULL 終端)
nBufKoe	koe バッファのサイズ[word]。バッファサイズ以上の音声記号列は切り捨てられるので入力テキストの2倍以上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_ConvRoman_sjis　　AqKanji2Koe.h

説明　漢字かな混じりのテキストを音声記号列(ローマ字)に変換(Shift-JIS 版)

構文　int **AqKanji2Koe_ConvRoman_sjis** (void * *hAqKanji2Koe-A*, const char **kanji*, char **koe*, int *nBufKoe*)

引数

hAqKanji2Koe-A	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（ShiftJIS, NULL 終端)
koe	出力バッファ。ローマ字音声記号列文字列が返る（ASCII, NULL 終端)
nBufKoe	koe バッファのサイズ[byte] 256 以上を指定。変換後の音声記号列がこのサイズを超えるとエラーを返すので、入力テキストの2倍以上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。

AqUsrDic.dll（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、異なるシステム辞書で生成したユーザ辞書は互換性が無い。

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

AqKanji2Koe-A 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。
AqUsrDic 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

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

サンプルプログラム

コード説明

次に示すコードは、日本語のテキストを標準入力から読み込んで、音声記号列を標準出力に出力するだけの単純なプログラムです(SDK パッケージ内の/samples/ Kanji2KoeCmd/Kanji2KoeCmd.cpp と基本的に同じものです)。

基本的な処理の流れは、最初に言語処理部のインスタンスを生成し（25 行目）、1 行毎にテキストを読み込んで音声記号列に変換して出力しています。最後の行を処理したら AqKanji2Koe_Release()でインスタンスを開放しています。

このプログラムで出力した音声記号列を AquesTalk に与えることで音声データに変換することができます。

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <Windows.h>
5   #include <AqKanji2Koe.h>   // インクルードパスの追加が必要
6   #pragma comment(lib, "AqKanji2Koe.lib")   // AqKanji2Koe-A ライブラリをリンク
7   #pragma warning(disable : 4996)
8
9   #define     NSTR  4096
10
11  void _AqGetExePath(char *path);
12
13  int main(int ac, char **av)
14  {
15      int iret;
16      char kanji[NSTR];
17      char koe[NSTR];
18      char dicDir[_MAX_PATH];
19
20      // 辞書データファイルのパスを求める
21      _AqGetExePath(dicDir);
22      strcat(dicDir, "aq_dic");
23
24      // 言語処理ライブラリのインスタンス生成
25      void *hAqKanji2Koe = AqKanji2Koe_Create(dicDir, &iret);
26      if(hAqKanji2Koe==0) // エラーの場合、0 が返る エラーコードは iret に
27          return iret;
28
29      int i;
30      for(i=0; ; i++){
31          if(fgets(kanji, NSTR, stdin)==0) break;
32          // 漢字かな交じりのテキストデータを音声記号列に変換
33          iret = AqKanji2Koe_Convert_sjis(hAqKanji2Koe, kanji, koe, NSTR);
34          if(iret!=0) break;   // エラーの場合、0 以外が返る
35          fprintf(stdout, "%s\n", koe);
36      }
37
38
```

```
39      // 言語処理ライブラリのインスタンス解放
40      AqKanji2Koe_Release(hAqKanji2Koe);
41      return iret;
42  }
```

AqKanji2Koe_Create()でインスタンスを生成する場合、辞書データは内部メモリに一旦すべて読み込まれます。

一方、AqKanji2Koe_Create_Ptr()では、呼び出し側で辞書データを読み込みます。例えば、サーバ利用などで複数の言語処理プロセスを起動する場合は、メモリマップトファイルを使って複数のプロセスで辞書データのメモリを共有することができます。

上記サンプルでは、AqKanji2Koe_Convert_sjis()を用いて、入出力に Shift-JIS の文字コードを使用しています。文字コードが UTF-8 の場合には、AqKanji2Koe_Convert_utf8()を使用します。また、Unicode 環境で開発される場合は、入出力が UTF16(char16_t)の AqKanji2Koe_Convert_utf16()を用いたほうが簡単です。なお、生成した音声記号列を AquesTalk に与える場合は、同じ文字コードの関数を用いてください。

C や C++以外の言語環境(たとえば C#,VB など）での使用方法はここでは示しません。データ型の変換等の処理が必要になるでしょう。詳細は各言語の DLL を呼び出す一般的な方法を参照してください。また、一部の関数はポインタを返しますので、VB ではラッパーが必要になるかもしれません。

ビルド

SDK パッケージにはサンプルアプリのプロジェクトが 1 つ含まれています。Kanji2KoeCmd は、上記サンプルと同じものです。

プロジェクトは VisualStudio 2017 用のプロジェクト設定ファイルが含まれていますので、これらの開発環境の場合はプロジェクトファイルを読み込んでビルドするだけでサンプルアプリの実行モジュールが生成できると思います。

これ以外の開発環境の場合は、サンプルアプリのソースコードを使ってビルド環境を開発環境に合わせて設定してください。

実行

実行時には、AqKanji2Koe.dll と辞書データを指定された場所に配置しておく必要があります。Kanji2KoeCmd サンプルプログラムでは exe と同じディレクトリに、AqKanji2Koe.dll と辞書ファイルのフォルダ(aq_dic)を配置してください。サンプルプログラムの配置は以下のとおりです。なお、システム辞書(aqdic.bin)とユーザ辞書(aq_user.dic)のファイル名は、システム固定であり変更できません。

```
|- Kanji2KoeCmd.exe
|- AqKanji2Koe.dll
|- aq_dic/
    |- aqdic.bin
    |- aq_user.dic (ユーザ辞書は必要に応じて)
```

コマンドプロンプトを起動し、Kanji2KoeCmd.exe を実行すると、入力待ち状態になります。ここで適当な文を入力すると、音声記号列が出力されます。

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。

AquesTalk.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	更新内容
2011/01/17	1.0	新規作成
2013/06/26	2.0	Ver.2 用に書き換え
2017/11/11	3.0	Ver.3 用に書き換え
2019/03/03	4.0	Ver.4 用に書き換え
2022/12/06	4.2	ローマ字音声記号列の関数追加