

PenPower Technology Ltd.
Taiwan

Hand-Written Recognition Engine API Document

Please keep in Confidential

(Preliminary Date : 12/23/2009)
(Revised Date : 10/08/2012)
(Revised Date : 07/15/2013)
Remove SKB APIs.
(Revised Date : 11/11/2013)
Remove Learn Parameters from HWRConnect.
3 Phrase APIs added.
(Revised Date : 03/03/2014)
APIs revised.
4 APIs added for Unicode manipulation.
(Revised Date : 12/03/2014)
2 character-set definitions added for Hiragana
and Katakana.
(Revised Date : 08/26/2024)
Modify PPHWRConnect API, 2 more parameters added.

Assumption:

The upper left corner of the Digitizer is the origin.

蒙恬科技股份有限公司

Data Structure and Constants Definition, Functions' ProtoType

```
typedef struct point_type {
    short  x;
    short  y;
} POINT_TYPE, *LPPOINT_TYPE;
```

```
#ifndef BYTE
#define BYTE    unsigned char
#endif
#ifndef WORD
#define WORD    unsigned short
#endif
```

```
#ifndef DWORD
#define DWORD    unsigned long
#endif
```

```
/* Following are the return value definition for the mandatory API functions */
#define STATUS_OK                0x0000 /* No Error */
#define STATUS_NO_DICT           0x0001 /* Miss Database Address */
#define STATUS_INVALID_DICT      0x0002 /* Database Format Error */
#define STATUS_INVALID_TYPE      0x0003 /* Unsupported Recognition Type */
#define STATUS_INVALID_NUMBER    0x0004 /* Candidate Count Invalid */
#define STATUS_NO_ENOUGH_MEMORY  0x0005 /* More Working RAM Required */
#define STATUS_INVALID_HWRDATA_PTR 0x0006 /* Invalid Object Pointer */
#define STATUS_INVALID_PRIVATE_PTR 0x0007 /* Invalid WorkBuffer Pointer */
#define STATUS_INVALID_BOX_PTR   0x0008 /* Invalid Rectangle Pointer */
#define STATUS_INVALID_TRACE_PTR 0x0009 /* Invalid Ink Buffer Pointer */
#define STATUS_INVALID_RESULT_PTR 0x000A /* Invalid Result Buffer Pointer */
#define STATUS_UNKNOWN           0x00FE /* Unknown Error */
```

```
/* The Gestures' Definitions */
```

```
//Delete Key on KeyBoard
```

```
#define DELETE_GESTURE    0x0010
```

```
//Return Key on KeyBoard
```

```
#define RETURN_GESTURE    0x000d
```

```
//Space Bar on KeyBoard
```

```
#define SPACE_GESTURE     0x0020
```

```
//BackSpace Key on KeyBoard
```

```
#define BACKSPACE_GESTURE 0x0008
```

// Following are the mandatory API functions
// APIs for HWRE JNI module

```
int PPHWRConnect(byte[] aHWRPath,  
                 boolean aAiEnable,  
                 byte[] aAiPath,  
                 boolean aAiMultiEnable,  
                 byte[] aAiMultiPath,  
                 boolean aDictEnable,  
                 byte[] aDictPath,  
                 boolean aLearnEnable,  
                 byte[] aLearnPath);  
int PPHWRSetType(int aType);  
int PPHWRSetBox(int aLeft,  
               int aTop,  
               int aRight,  
               int aBottom);  
int PPHWRSetCandidateNum(short aNumber);  
int PPHWRRecognize(short[] aTrace,  
                  char[] aResult);  
int PPHWRSetResolution(int width,  
                      int height);  
int PPHWRProcessStrokes(short[] aTrace);  
int PPHWRFinalProcess(char[] aResult);  
int PPHWRProcessOverlap(short[] aTrace,  
                       short aEndStrokeIndex,  
                       char[] aResult,  
                       short aOutputFormat);  
int PPHWRFinalProcessOverlap();  
int PPHWROverlapWriteSegmentInk(short[] aTrace,  
                               short start_x,  
                               short start_y,  
                               char[] aResult,  
                               char[] aSegmentInkResult);  
short PPHWRConvertUTF16_2_UTF32(short[] autf16_str,  
                                int[] autf32_str,  
                                int no_entries);  
short PPHWRConvertUTF32_2_UTF16(int[] autf32_str,  
                                short[] autf16_str,  
                                int no_entries);  
int PPHWRUTF32_2_Surrogate (int[] aResultU32,  
                           short[] aResultU16,  
                           int U32count);  
int PPHWRSurrogate_2_UTF32 (short [] aResultU16,  
                           int[] aResultU32,  
                           int U16count);
```

```
void PPHWRDisconnect();  
int PPHWRRecogSentence_Eng(short[] aTrace,  
                           int Mode,  
                           char[] aResult);
```

蒙住科技股份有限公司

API Definitions:

1. `int PPHWRConnect(byte[] aHWRPath,
 boolean aAiEnable,
 byte[] aAiPath,
 boolean aAiMultiEnable,
 byte[] aAiMultiPath,
 boolean aDictEnable,
 byte[] aDictPath,
 boolean aLearnEnable,
 byte[] aLearPath);`

INPUT : (1) path for Internal Dict.
(2) Enable/Disable switch for AI.
(3) path for AI Dict.
(4) Enable/Disable switch for Learn.
(5) path for Learn Dict.
(6) Enable/Disable switch for Alphabets AI.
(7) path for Alphabets AI Dict.
(8) Enable/Disable switch for Alphabets User Dict.
(9) path for Alphabets User Dict.
(10) Enable/Disable switch for Learn Dict.
(11) path for Learn Dict.

OUTPUT : return 0 : indicate connection was built successfully.
return nonzero : indicate connection error.

DETAIL : Before any other Hand-writing recognition API called,
this function must be called to make engine work well.
The first parameter, "path for Internal Dict" cannot be
an empty or NULL string; others unused parameters
just setup to NULL string and FALSE.

2. int PPHWRSetType(int aType)*

INPUT : The specified recognition types.

OUTPUT : return 0 : indicate no error detected.
return nonzero indicate error found.

DETAIL : Default value: All types.

/* Following are Recognition Type Definition */

TYPE	VALUE	MEANING
ALL_TYPE	NO_NEED	Recognize all supported Subsets (HK not included)
SYMBOL_TYPE	NO_NEED	Supported Symbols
ALPHA_TYPE	NO_NEED	Supported English Alphabets
NUMERIC_TYPE	NO_NEED	Supported Numbers
HK_TYPE	NO_NEED	HK Character
SIMPLIFIED_ONLY_TYPE	NO_NEED	Convert Chinese Character to its Simplified form.
TRADITIONAL_ONLY_TYPE	NO_NEED	Convert Chinese Character to its Traditional form.
GESTURE_TYPE	NO_NEED	GESTURE
CHINESE_TYPE	NO_NEED	Chinese Character

Examples:

(1) **All supported Characters, Just as writing form:**

PPHWRSetType(ALL_TYPE+HK_TYPE); //Gestures **included**

PPHWRSetType(ALL_TYPE+HK_TYPE-GESTURE_TYPE); //Gestures **not included**

(2) **All supported Characters, convert to Simplified Chinese form:**

PPHWRSetType(ALL_TYPE+SIMPLIFIED_ONLY_TYPE+HK_TYPE);

PPHWRSetType(ALL_TYPE+SIMPLIFIED_ONLY_TYPE+HK_TYPE
-GESTURE_TYPE);

(3) **All supported Characters, convert to Traditional Chinese form:**

PPHWRSetType(ALL_TYPE+TRADITIONAL_ONLY_TYPE+HK_TYPE);

PPHWRSetType(ALL_TYPE+TRADITIONAL_ONLY_TYPE+HK_TYPE
-GESTURE_TYPE);

(4) **Support only Chinese Characters, Just as writing form:**

PPHWRSetType(ALL_TYPE+HK_TYPE-SYMBOL_TYPE-ALPHA_TYPE-
NUMERIC_TYPE);

PPHWRSetType(ALL_TYPE+HK_TYPE-SYMBOL_TYPE-ALPHA_TYPE-
NUMERIC_TYPE-GESTURE_TYPE);

(5) Support only Chinese Characters, convert to Simplified Chinese form:

```
PPHWRSetType(ALL_TYPE+HK_TYPE -SYMBOL_TYPE-ALPHA_TYPE-  
              NUMERIC_TYPE+SIMPLIFIED_ONLY_TYPE);
```

```
PPHWRSetType(ALL_TYPE+HK_TYPE -SYMBOL_TYPE-ALPHA_TYPE-  
              NUMERIC_TYPE-GESTURE_TYPE+SIMPLIFIED_ONLY_TYPE);
```

(6) Support only Chinese Characters, convert to Traditional Chinese form:

```
PPHWRSetType(ALL_TYPE+HK_TYPE -SYMBOL_TYPE-ALPHA_TYPE-  
              NUMERIC_TYPE+TRADITIONAL_ONLY_TYPE);
```

```
PPHWRSetType(ALL_TYPE+HK_TYPE -SYMBOL_TYPE-ALPHA_TYPE-  
              NUMERIC_TYPE-GESTURE_TYPE+  
              TRADITIONAL_ONLY_TYPE);
```

***NOTE**

1. : To improve the Hit Rate, ASCII_TYPE(Alphabets, Symbols, Numbers) are not included in ALL_TYPE; that is, Chinese characters recognition and ASCII characters **CANNOT** be recognized at the same time. Your App must turn on either ALL_TYPE or ASCII_TYPE to get Chinese characters or ASCII characters in two different API calls.

3. int PPHWRSetBox(int aLeft, int aTop, int aRight, int aBottom)

INPUT : Rectangle coordinates to bound the writing pad.

OUTPUT : return 0 : indicate no error detected.
return nonzeror : indicate error found.

DETAIL : If the recognition types are set to the combination of Lower case ,Upper case alphabets, and symbols, the setting will influence the judgement for such as "C" and "c", "S" and "s", "" and ",", etc, their shapes are the same but different in size. Default value is (0,0,60,60).

4. int PPHWRSetCandidateNum(short aNumber)

INPUT : The max expected candidate no.

OUTPUT : return 0 : indicate no error detected.
return nonzeror : indicate error found.

DETAIL : The HWRE will return candidates up to the limit number. The valid value is 1~15. Default value is 10.

5. int PPHWRRecognize(short[] aTrace, char[] aResult);

INPUT : Trace buffer pointer,
result buffer pointer.

OUTPUT : return 0 : indicate no error detected.
return nonzeror : indicate error found.

DETAIL : (1) The AP must not check the content of result buffer unless the error code is equal to 0 (no error detected).
(2) The result buffer must have (n+1) entries.
where n is the max expected returned candidate count.
Since the HWRE will put a "0" at the end of candidate list to mark the search end, instead of return candidate count.

(3) This function is used while UTF16 encoding required.

6. int PPHWRSetResolution(int width,
 int height);

INPUT : Width and Height of Writing Area.

OUTPUT : return 0 : indicate no error detected.
 return nonzeror : indicate error found.

DETAIL : Call this function will affect the judgement
 for Upper/Lower case alphabets and some
 symbols(such as : “.”, “;”, “”). It also change
 the parameters for Continuous Chinese Writing.
 Default value is (640 x 480).

蒙恬科技股份有限公司

7. int PPHWRProcessStrokes(short[] aTrace);

INPUT : Buffer pointer for User input Ink data.

OUTPUT : return 0 : indicate no error detected.
return nonzeror : indicate error found.

DETAIL : This function is used to collect and keep ink data for HWRE while continuous writing function required. It won't return the intermediate result.

8. int PPHWRFinalProcess(char[] aResult);

INPUT : Pointer for candidate array.

OUTPUT : A positive value indicate recognized chars,
A negative value indicate error found.

DETAIL : This function is used to retrieve the final result of continuous writing. Result array will be composed as :
offset 0 : number of characters found.
offset 2; number of candidates for first char.
offset 4: First candidate Unicode of first char.
offset 6; Second candidate Unicode of first char.
... < repeat from offset 2 if all candidates for first char parsed>

offset last char parsed : 0. It means no more chars.
NOTE : (1) This function is used while UTF16 encoding required.

9. int PPHWRProcessOverlap(short[] aTrace,
 short aEndStrokeIndex,
 char[] aResult,
 short aOutputFormat);

- INPUT : Buffer pointer for User input Ink data,
 Index of last stroke to recognize, start from 0,
 Pointer for candidate array,
 Indicate result buffer format.
- OUTPUT : return 0 : indicate no error detected.
 return non-zero : indicate error found.
- DETAIL : This function is used to runtime recognize overlapped
 writing ink
- NOTE : (1) The AP should not use the returned Result array,
 if returned error code not equal to STATUS_OK.
 (2) The structure for result buffer is :
 when OutputFormat=0
 aResult+0 : candidate string1 length (len1)
 aResult+1 : candidate string1.
 aResult+1+len1 : candidate string2 length (len2)
 aResult+1+len1+1 : candidate string2.
 <etc>
 aResult +n : 0, end mark
 when OutputFormat=1
 aResult+0 : candidate string1 length (len1)
 aResult+1 : candidate string1.
 aResult+1+len1 : stroke count for each char in candidate
 string1.
 aResult+1+len1*2 : candidate string2 length (len2)
 aResult+1+len1*2+1 : candidate string2.
 aResult+1+len1*2+1+len2 : stroke count for each char in
 candidate string2.
 <etc>
 aResult +n : 0, end mark
 (3) aResult is the same memory buffer for
 PPHWROverlapWriteSegmentInk parameter 4. Before
 writing a new overlap writing ink, the content of this
 buffer must be set to 0.
 (4) Size of aResult is proposed to be 4096.

10. int PPHWRFinalProcessOverlap();

- OUTPUT : return 0 : indicate no error detected.
return nonzero : indicate error found.
- DETAIL : This function is used to clean up recognition variables
- NOTE : (1) Without calling this functions, the recognition variables will not be cleaned up and this will cause problems for new input sessions.

11. int PPHWROverlapWriteSegmentInk(short[] aTrace,
short start_x,
short start_y,
char[] aResult,
char[] aSegmentInkResult);

- INPUT : Buffer pointer for User input Ink data,
Next Stroke Pen-Down Start Point x coordinate,
Next Stroke Pen-Down Start Point y coordinate,
Pointer for candidate array,
Pointer for information array of new component.
- OUTPUT : return 0 : indicate no error detected.
return non-zero : indicate error found.
- DETAIL : This function is used to detect stroke component for display ink.
- NOTE : (1) The AP should not use the returned result, if returned error code not equal to STATUS_OK.
(2) pResult is the same memory buffer for PPHWRProcessOverlap parameter 3 or PPHWROverlapWrite parameter 2. Before writing a new overlap writing ink, the content of this buffer must be set to 0.
(3) Size of aResult is proposed to be 4096.
(4) aSegmentInkResult [0] indicate stroke begin index of new component, aSegmentInkResult [1] indicate stroke end index of new component

12. short PPHWRConvertUTF16_2_UTF32 (short[] autf16_str,
int[] autf32_str,
int no_entries);

INPUT : Pointer for UTF16 candidate array,
Pointer for UTF32 candidate array,
elements count of UTF16.

OUTPUT : The result for Convert.
0 – success.
other – fail.

DETAIL : Use this function as AP supporting UTF32 gets returned
candidate list to convert formatted UTF16 string to UTF32
string.

13. short PPHWRConvertUTF32_2_UTF16(int[] autf32_str,
short[] autf16_str,
int no_entries);

INPUT : Pointer for UTF32 candidate array,
Pointer for UTF16 candidate array,
elements count of UTF32.

OUTPUT : The result for Convert.
0 – success.
other – fail.

DETAIL : Use this function before AP supporting UTF32 pass
string to recognition engine to convert the string to UTF16
string.

14. int PPHWRUTF32_2_Surrogate (int[] aResultU32,
char[] aResultU16,
int U32count);

INPUT : Pointer for UTF32 candidate array,
Pointer for UTF16 candidate array,
elements count of UTF32.

OUTPUT : A positive value indicate converted successfully,
A negative value indicate error found.

DETAIL : Generally speaking, the array, aResultU16, should be
the same size in bytes compared with aResultU32.
It will try to convert a UTF32 0x10000~0x10FFFF
into 2 composited UTF16 code, LoWord first, then
followed the HiWord. LoWord will be in Range
0xDC00~0xDFFF. Once the AP parses the content
aResultU16 and find a code in this range, it should read
one more Word to finish to composition.
**The buffer size of aResultU16 must be (U32count+1)*2
char at least.**
**If your system can handle UTF32 code, your AP does
not need to call this function.**

15. int PPHWRSurrogate_2_UTF32 (char[] aResultU16,
int[] aResultU32,
int U16count);

INPUT : Pointer for UTF16 candidate array,
Pointer for UTF32 candidate array,
Candidate count of UTF16.

OUTPUT : A positive value indicate converted successfully,
A negative value indicate error found.

DETAIL : Generally speaking, the array, aResultU16, should be
the same size in bytes compared with aResultU32.
It is the reverse operation of function
“PPHWRConvertU32ToSurrogate”.
**The buffer size of aResultU32 must be (U16count +1) int at
least.**
**If your system can handle UTF32 code, your AP does
not need to call this function.**

16. void PPHWRDisconnect()

INPUT : NONE.

OUTPUT : NONE.

DETAIL : Release allocated resources for Hand-write recognition engine. AP must call this function before quit to prevent from resource leak or make sure resource released before a new connection built.

17. int PPHWRRecogSentence_Eng(short[] aTrace,
int Mode,
char[] aResult,
Object aContext);

INPUT : Trace buffer pointer,
Indicate sentence or word mode,
result buffer pointer,
context.

OUTPUT : return <=0 : indicate error found.
return >0 : indicate no error detected.

DETAIL : (1) The AP must not check the content of result buffer
unless the error code is > 0 (no error detected).
(2) Mode =1 indicate sentence mode, Mode=2 indicate
word mode.

NOTE : (1) The structure for result buffer is :
when Mode=1,
aResult+0 : line count
{
aResult+1 : left of line 1 rectangle
aResult+2 : top of line 1 rectangle
aResult+3 : right of line 1 rectangle
aResult+4 : bottom of line 1 rectangle
aResult+5 : word count in line 1
{
aResult+6 : left of word 1 rectangle
aResult+7 : top of word 1 rectangle
aResult+8 : right of word 1 rectangle
aResult+9 : bottom of word 1 rectangle
aResult+10 : stroke count of word 1 (n)
aResult+10+1~10+n : stroke index of word 1


```

aResult+10+n+1 : char count in word 1
{
aResult+10+n+2 : left of char 1 rectangle
aResult+10+n+3 : top of char 1 rectangle
aResult+10+n+4 : right of char 1 rectangle
aResult+10+n+5 : bottom of char 1
                rectangle
aResult+10+n+6 : code of char 1
//content for char 2
<etc>
}
//content for word 2
<etc>
}
//content for line 2
<etc>
}

```

```

when Mode=2,
string 1 with end mark 0,
0xfffe,
string 2 with end mark 0,
0xfffe,
string 3 with end mark 0,
0xfffe,
<etc>
string N with end mark 0,
0xffff.

```

Supplements:

1. : Realtime Inking:

Since only "HWRRecognize" will take at most 1 sec. to get response, if AP would like to use two or above writing-boxes to input, it must run in at least two threads/tasks:
UI thread(for inking) and Working thread(for recognition).

2. : Re-Entrant and Memory recycling:

Inside a process, the package is NOT Re-Entrant for different threads.

3. : Symbols/Numerics/Alphabets are in their ASCII form instead of MBCS form, that is, "A"(0x41) will be returned as 0x0041 instead of 0x4100.

4. : Suggested flow to invoke the HWRE:

- (1) HWRConnect -- prepare resources for recognition engine.
- (2) HWRSetType/HWRSetBox/HWRSetCandidateNum
-- These four functions' order is arbitrary.
- (3) HWRRecognize -- As the AP receives a "Recognize!" event via "Timer", "Button press", "box switch", it call this function to get result.
- (4) return to step (2) until the HWRE can be released.
- (5) HWRDisconnect – release resource allocated for recognition.

5. : The ink data structure and "end mark"

The AP should store the collected ink data in its own buffer until "PPHWRRecognize" called.

Once AP receive a stroke data(WM_LBUTTONDOWN or PEN_UP event), it must add an "end-mark",(-1,-1), at the end of this stroke data.

However, before send call "PPHWRRecognize", AP should change the end mark of last stroke to (-1,0).

Example 1.

(10,10),(20,10),(30,10),(-1,0)

parameter "aTrace" will has such a form:

10,10,20,10,30,10,-1,0.

Example 2.

(10,10),(20,10),(30,10),(-1,-1),(10,20),(20,20),(30,20),(-1,0)

parameter "aTrace" will has such a form:

10,10,20,10,30,10,-1,-1,10,20,20,20,30,20,-1,0.

6. : As "HWRRecognize" returns 0, then the AP can check the content point by aResult. Each non-zero WORD is a candidate, the most possible candidate is the first, and the least possible one is the last; "0" means "no more candidate" available. For Example:
(*pResult)[]={0xA440, 0xA441, 0xA442, 0} :
means "Three" candidates are returned.
Their codes are :
0xA440,0xA441, and 0xA442.

- 7: Default Gestures(Special commands for AP)
As turned the correspondent GESTURE_TYPE bit on in Recognition type setting, the HWRE will return default gestures as candidate. So called “Default Gesture” is a special command that AP must take some actions predefined while the codes of Default Gestures are found in candidate list. The followings are the writing scripts, codes, and actions must be taken for these Default Gestures:



: Backspace_Gesture, Backspace, delete previous character from current cursor position.



: Delete_Gesture, Delete, delete character at current cursor position.



: Return_Gesture, Return, insert a “Return” character at current cursor position.



: Space_Gesture, Space, insert a “Space” character at current cursor position.

The black dot is just used to show where to start the script.