libhandwriting Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Constants

Number and Size of Data

Number of data and data size used by libhandwriting

Definition

Value	Decimal	Description
SCE_HANDWRITING_MAX_CANDIDATES	32	Maximum value of recognition result items
SCE_HANDWRITING_MAX_STROKE	31	Maximum number of strokes of handwriting data
SCE_HANDWRITING_MAX_POINTS	1024	Maximum number of handwriting data items
SCE_HANDWRITING_MAX_REGISTER_COUNT	32767	Maximum value of the number of registered characters
SCE_HANDWRITING_REGISTER_BLKSIZE	512	Memory size required per registered character
SCE_HANDWRITING_MAX_REGISTER_CANDIDATES	32	Maximum number of character recognition candidates that can be registered

Description

These constants define the maximum values and fixed values used by libhandwriting.

Dictionary Number

Dictionary number for initialization

Definition

Value	Decimal	Description
SCE_HANDWRITING_DIC_ALNUM	0	Alphanumeric characters
SCE_HANDWRITING_DIC_ALNUM_NATIVE	1	Alphanumeric characters (westernized) The shape of the characters is the same as SCE_HANDWRITING_DIC_ALNUM, but the stroke order has been optimized for Western countries.
SCE_HANDWRITING_DIC_HIRA	2	Hiragana
SCE_HANDWRITING_DIC_KATA	3	Katakana
SCE_HANDWRITING_DIC_KANJI	4	Kanji
SCE_HANDWRITING_DIC_KANJI2	5	Kanji 2
SCE_HANDWRITING_DIC_ALL	6	All character types (Japan)
SCE_HANDWRITING_DIC_ALL2	7	All character types (Japan) 2
SCE_HANDWRITING_DIC_ALNUMLAT1	8	Latin 1
SCE_HANDWRITING_DIC_KOREAN	9	Korea
SCE_HANDWRITING_DIC_HANGUL	10	Hangul
SCE_HANDWRITING_DIC_GERMAN	11	Germany
SCE_HANDWRITING_DIC_DUTCH	12	Holland
SCE_HANDWRITING_DIC_FRENCH	13	France
SCE_HANDWRITING_DIC_ITALIAN	14	Italy
SCE_HANDWRITING_DIC_SPANISH	15	Spain
SCE_HANDWRITING_DIC_PORTUGUESE	16	Portugal
SCE_HANDWRITING_DIC_RUSSIAN	17	Russia
SCE_HANDWRITING_DIC_DAN_NOR	18	Denmark, Norway
SCE_HANDWRITING_DIC_SWE_FIN	19	Sweden, Finland
SCE_HANDWRITING_DIC_ICELANDIC	20	Iceland
SCE_HANDWRITING_DIC_POLISH	21	Poland
SCE_HANDWRITING_DIC_SIMPLIFIED_CHINESE	22	Simplified Chinese
SCE_HANDWRITING_DIC_TRADITIONAL_CHINESE	23	Traditional Chinese

Description

These are the dictionary number constants used by `sceHandwritingGetBufferSize()` and `sceHandwritingInit()`.

Recognition Mode

Constants specified for the recognition mode

Definition

Value	Decimal	Description
SCE_HANDWRITING_MODE_SYMBOL	0x0001	Symbols
SCE_HANDWRITING_MODE_NUMBER	0x0002	Numeric characters
SCE_HANDWRITING_MODE_UALPHA	0x0004	Uppercase English characters
SCE_HANDWRITING_MODE_LALPHA	0x0008	Lowercase English characters
SCE_HANDWRITING_MODE_HIRA	0x0010	Hiragana
SCE_HANDWRITING_MODE_KATA	0x0020	Katakana
SCE_HANDWRITING_MODE_GREEK	0x0040	Greek characters
SCE_HANDWRITING_MODE_HANGUL	0x0080	Hangul
SCE_HANDWRITING_MODE_HANJA	0x0100	Korean Hanja
SCE_HANDWRITING_MODE_KANJI1	0x0400	JIS 1st level Kanji GB2312-80 1st level Kanji
SCE_HANDWRITING_MODE_KANJI2	0x0800	JIS 2nd level Kanji GB2312-80 2nd level Kanji
SCE_HANDWRITING_MODE_URUSSIAN	0x0080	Uppercase Russian characters
SCE_HANDWRITING_MODE_LRUSSIAN	0x0100	Lowercase Russian characters
SCE_HANDWRITING_MODE_HONGKONG	0x0080	Hong Kong Supplementary Character Set (HKSCS)
SCE_HANDWRITING_MODE_ALL	0x1FFF	All character recognition

Description

These are the recognition mode constants used by `sceHandwritingSetMode()`.

Return Codes

List of return codes returned by libhandwriting

Definition

Value	Hexadecimal	Description
SCE_HANDWRITING_ERROR_INVALID_POINTER	0x80105600	Invalid argument pointer
SCE_HANDWRITING_ERROR_INVALID_PARAM	0x80105601	Invalid argument value
SCE_HANDWRITING_ERROR_NOT_OPENED	0x80105602	Could not open dictionary file
SCE_HANDWRITING_ERROR_ALREADY_INITIALIZED	0x80105603	Initialization has already been performed
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_READ	0x80105605	Failed to read a file
SCE_HANDWRITING_ERROR_MEMORY_ERROR	0x80105606	Memory error
SCE_HANDWRITING_ERROR_DIC_FULL	0x80105607	No more characters can be registered
SCE_HANDWRITING_ERROR_CODE_ERROR	0x80105608	Invalid character code
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Datatypes

SceHandwritingPoint

Coordinate information of handwriting data

Definition

```
#include <libhandwriting/libhandwriting_api.h>
typedef struct SceHandwritingPoint
{
        SceUChar8 x;
        SceUChar8 y;
} SceHandwritingPoint;
```

Members

x Coordinate axis X
y Coordinate axis Y

Description

This datatype is used for handwriting information storage of handwriting data. It is used by `sceHandwritingRecognize()`, `sceHandwritingRegisterSet()`, and `sceHandwritingRegisterGetResult()`.

Coordinates *x* and *y* are valid in the range of 0 to 127. The most significant bit of the *x* coordinate serves as the pen up/pen down flag. In the case of coordinates (0x14, 0x18), (0x94, 0x18) is saved in the case of pen down, and (0x14, 0x18) is saved in the case of pen up.

Pen down state must be maintained during one stroke except for the end of the stroke. The end of each stroke must be pen up.

If both the *x* and *y* coordinates are 0xFF, this indicates that this is the end of the data.

See Also

`sceHandwritingRecognize()`, `sceHandwritingRegisterSet()`, `sceHandwritingRegisterGetResult()`

SceHandwritingData

Handwriting data and candidate storage structure

Definition

```
#include <libhandwriting/libhandwriting_api.h>
typedef struct SceHandwritingData
{
        SceWChar32 codes[SCE_HANDWRITING_MAX_CANDIDATES];
        SceHandwritingPoint points[SCE_HANDWRITING_MAX_POINTS];
} SceHandwritingData;
```

Members

codes	Character code array for storing recognition results
points	Coordinate information array of handwriting data

Description

This datatype is used to set the handwriting data and receive the recognition results by calling the recognition function. It is used by `sceHandwritingRecognize()`.

Maximum Data

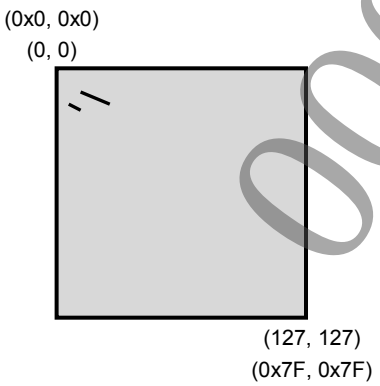
- The maximum number of coordinates per character is the value defined by `SCE_HANDWRITING_MAX_POINTS`, but since (0xFF, 0xFF), which indicates the data end, is required, the actual number is `SCE_HANDWRITING_MAX_POINTS` -1.
- The maximum number of coordinate points that can be recognized per stroke is 256. If there are more than 256 coordinate points, cull processing is performed.
- The maximum number of strokes (input strokes) is 31. Strokes in excess of this number are ignored.

When the X and Y coordinate values are input in the following example, where the top left of the input frame is (0,0), the contents saved to *points* of the structure are as follows.

Example: Coordinate points sequence

1st stroke (0x15, 0x18), (0x18, 0x19), (0x1b, 0x1b)

2nd stroke (0x10, 0x19), (0x14, 0x1a)



points[0].x: 0x95: 1st stroke 1st point x coordinate +0x80 (pen down)
points[0].y: 0x18: 1st stroke 1st point y coordinate
points[1].x: 0x98: 1st stroke 2nd point x coordinate +0x80 (pen down)
points[1].y: 0x19: 1st stroke 2nd point y coordinate
points[2].x: 0x1b: 1st stroke third point x coordinate (pen up)
points[2].y: 0x1b: 1st stroke third point y coordinate
points[3].x: 0x90: 2nd stroke 1st point x coordinate +0x80 (pen down)
points[3].y: 0x19: 2nd stroke 1st point y coordinate
points[4].x: 0x14: 2nd stroke 2nd point x coordinate (pen up)
points[4].y: 0x1a: 2nd stroke 2nd point y coordinate
points[5].x: 0xFF: End mark
points[5].y: 0xFF: End mark

See Also

`sceHandwritingRecognize()`, `SceHandwritingPoint`

SceHandwritingPointsData

Coordinate information structure of handwriting data

Definition

```
#include <libhandwriting/libhandwriting_api.h>
typedef struct SceHandwritingPointsData
{
        SceHandwritingPoint points[SCE_HANDWRITING_MAX_POINTS];
} SceHandwritingPointsData;
```

Members

points Coordinate information array of handwriting data

Description

This datatype is used to set the handwriting data and call the character registration function. It is used by `sceHandwritingRegisterSet()`.

See Also

`sceHandwritingRegisterSet()`,`SceHandwritingPoint`

SceHandwritingRegisterData

Character code array structure for storing results in registered dictionary

Definition

```
#include <libhandwriting/libhandwriting_api.h>
typedef struct SceHandwritingRegisterData
{
        SceWChar32 codes[SCE_HANDWRITING_MAX_REGISTER_CANDIDATES];
} SceHandwritingRegisterData;
```

Members

codes Character code array that returns result of only registered characters in recognition results

Description

This datatype is used to obtain the results of only the registered characters in the recognition results.

It is used to obtain the results in the registered dictionary by using the `sceHandwritingRegisterGetResult()` function when `sceHandwritingRecognize()` ends normally.

See Also

`sceHandwritingRecognize()`,`sceHandwritingRegisterGetResult()`

libhandwriting Functions

sceHandwritingGetBufferSize

Get buffer size required for specified dictionary

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingGetBufferSize (
        SceUInt32 dic
);
```

Calling Conditions

Multithread safe.

Arguments

dic Value of dictionary whose required buffer size is to be obtained (SCE_HANDWRITING_DIC_ALNUM, etc.)
* For the values of the other dictionaries, refer to the "Dictionary Number" section.

Return Values

Returns the required buffer size. Returns a negative value for errors.

Value	Hexadecimal	Description
Positive value	-	Required buffer size (normal termination)
SCE_HANDWRITING_ERROR_INVALID_PARAM	0x80105601	Invalid argument value
SCE_HANDWRITING_ERROR_NOT_OPENED	0x80105602	Could not open the dictionary file

Description

This function gets the required buffer size for the specified dictionary. The application allocates the buffer size corresponding to the return value and passes it to the initialization function.

See Also

sceHandwritingInit()

sceHandwritingInit

Initialize libhandwriting

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingInit (
        SceUChar8 maxX,
        SceUChar8 maxY,
        SceUInt32 dic,
        ScePVoid dicBuffer
);
```

Calling Conditions

Multithread safe.

Arguments

maxX	Width of input frame (can be specified from 50 to 127)
maxY	Height of input frame (can be specified from 50 to 127)
dic	Dictionary for initialization For the values that can be specified, refer to the arguments of `sceHandwritingGetBufferSize()`.
dicBuffer	Pointer to memory area of the buffer size obtained with `sceHandwritingGetBufferSize()`

Return Values

Returns a negative value for errors.

Value	Hexadecimal	Description
SCE_OK	0x0	Initialization was successful (normal termination)
SCE_HANDWRITING_ERROR_INVALID_POINTER	0x80105600	Argument *dicBuffer* is invalid
SCE_HANDWRITING_ERROR_INVALID_PARAM	0x80105601	Invalid argument value
SCE_HANDWRITING_ERROR_NOT_OPENED	0x80105602	Could not open the dictionary file
SCE_HANDWRITING_ERROR_ALREADY_INITIALIZED	0x80105603	Initialization has already been performed
SCE_HANDWRITING_ERROR_READ	0x80105605	Failed to read a file
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

The application allocates a buffer of the size obtained with `sceHandwritingGetBufferSize()` and performs initialization with this function.

The memory of *dicBuffer* need not be zero-cleared.

If `sceHandwritingInit()` end normally, do not perform access to the memory area of *dicBuffer* until `sceHandwritingTerm()` is called.

See Also

`sceHandwritingGetBufferSize(),sceHandwritingTerm()`

sceHandwritingTerm

Terminate libhandwriting

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingTerm ();
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns a negative value for errors.

Value	Hexadecimal	Description
SCE_OK	0x0	Normal termination
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function is called to terminate libhandwriting.

See Also

sceHandwritingInit()

sceHandwritingRecognize

Cause recognition by libhandwriting

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingRecognize (
        SceHandwritingData *data
);
```

Calling Conditions

Multithread safe.

Arguments

data Pointer to the structure where handwriting data is stored

Return Values

Returns the number of recognized character codes upon normal termination. Returns a negative value for errors.

Value	Hexadecimal	Description
Value of 0 or greater	-	Number of recognized character codes (normal termination) 0 to SCE_HANDWRITING_MAX_CANDIDATES
SCE_HANDWRITING_ERROR_INVALID_POINTER	0x80105600	Argument *data* is invalid
SCE_HANDWRITING_ERROR_INVALID_PARAM	0x80105601	Invalid argument value
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function performs recognition based on the specified handwriting data. Upon normal termination of the processing, the number of recognized character codes is returned as the return value and the list of character codes (Unicode UCS-4) is stored to the *codes* member of the SceHandwritingData structure. The storing order is from the highest recognition assessment.

In the case that SCE_HANDWRITING_ERROR_INVALID_PARAM returns, the possible causes are as follows.

- The last coordinates of the handwriting data does not constitute the pen up information. In short, the most significant bit of the x coordinate is not 0.
- The value of (0xFF, 0xFF), which indicates the end of the data, does not exist in the end of the handwriting data.

See Also

sceHandwritingInit(),SceHandwritingData

sceHandwritingSetMode

Change recognition mode of libhandwriting

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingSetMode (
        SceUInt32 mode
);
```

Calling Conditions

Multithread safe.

Arguments

mode Recognition mode (SCE_HANDWRITING_MODE_SYMBOL, etc.)
* For the other constants, refer to the "Recognition Mode" section.

Return Values

Returns a negative value for errors.

Value	Hexadecimal	Description
SCE_OK	0x0	The recognition mode was successfully changed (normal termination)
SCE_HANDWRITING_ERROR_INVALID_PARAM	0x80105601	Invalid argument value
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function changes the recognition mode of libhandwriting. This function is used to obtain results, such as "numeric characters only" or "English characters only". After this function is executed, only the specified category is returned for the following sceHandwritingRecognize() function calling.

Example: To have only uppercase and lowercase English characters recognized

```
sceHandwritingSetMode(SCE_HANDWRITING_MODE_UALPHA |
SCE_HANDWRITING_MODE_LALPHA)
```

Upon normal termination of sceHandwritingInit(), SCE_HANDWRITING_MODE_ALL (recognition of all characters) is automatically selected as the recognition mode.

The characters targeted for recognition in the specified recognition mode depend on the selected dictionary.

* For A, B, E, Z, H, I, K, M, N, O, P, T, X among Greek uppercase characters, the character codes (Unicode UCS-4) for uppercase alphabetic characters are returned.

Dictionaries and Corresponding Recognition Modes

The various dictionaries and their corresponding recognition modes are listed in the table below.

	SCE_HANDWRITING_DIC_ALNUM Alphanumerics	SCE_HANDWRITING_DIC_ALNUM_NATIVE Alphanumerics (westernized)	SCE_HANDWRITING_DIC_HIRA Hiragana
SCE_HANDWRITING_MODE_NUMBER	10 numeric characters	10 numeric characters	Invalid
SCE_HANDWRITING_MODE_UALPHA	26 alphabetic uppercase characters	26 alphabetic uppercase characters	Invalid
SCE_HANDWRITING_MODE_LALPHA	26 alphabetic lowercase characters	26 alphabetic lowercase characters	Invalid
SCE_HANDWRITING_MODE_SYMBOL	, . ' " − ! ? 7 characters	, . ' " − ! ? 7 characters	、 。 ? ! ― 「 」 " ° 9 characters
SCE_HANDWRITING_MODE_HIRA	Invalid	Invalid	83 Hiragana characters
SCE_HANDWRITING_MODE_KATA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_GREEK	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI1	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI2	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANGUL	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANJA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	Invalid

	SCE_HANDWRITING_DIC_KATA Katakana	SCE_HANDWRITING_DIC_KANJI Kanji	SCE_HANDWRITING_DIC_KANJI2 Kanji 2
SCE_HANDWRITING_MODE_NUMBER	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_UALPHA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_LALPHA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_SYMBOL	、 。 ? ! ― 「 」 " ° 9 characters	仝 々 〆 3 characters	仝 々 〆 3 characters
SCE_HANDWRITING_MODE_HIRA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KATA	86 Katakana characters	Invalid	Invalid
SCE_HANDWRITING_MODE_GREEK	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI1	Invalid	2965 Kanji characters	2965 Kanji characters
SCE_HANDWRITING_MODE_KANJI2	Invalid	776 Kanji characters	3390 Kanji characters
SCE_HANDWRITING_MODE_HANGUL	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANJA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	Invalid

	SCE_HANDWRITING_DIC_ALL All character types dictionary	SCE_HANDWRITING_DIC_ALL2 All character types dictionary	SCE_HANDWRITING_DIC_ALNUMLAT1 Latin 1
SCE_HANDWRITING_MODE_NUMBER	10 numeric characters	10 numeric characters	10 numeric characters
SCE_HANDWRITING_MODE_UALPHA	26 alphabetic uppercase characters	26 alphabetic uppercase characters	26 alphabetic uppercase characters 33 Latin uppercase characters
SCE_HANDWRITING_MODE_LALPHA	26 alphabetic lowercase characters	26 alphabetic lowercase characters	26 alphabetic lowercase characters 34 Latin lowercase characters
SCE_HANDWRITING_MODE_SYMBOL	103 symbols	, . ' " – ! ? 7 characters	, . ' " – ! ? ¡ ¿ 9 characters
SCE_HANDWRITING_MODE_HIRA	83 Hiragana characters	83 Hiragana characters	Invalid
SCE_HANDWRITING_MODE_KATA	86 Katakana characters	86 Katakana characters	Invalid
SCE_HANDWRITING_MODE_GREEK	35 Greek uppercase and lowercase characters	35 Greek uppercase and lowercase characters	Invalid
SCE_HANDWRITING_MODE_KANJI1	2965 Kanji characters	2965 Kanji characters	Invalid
SCE_HANDWRITING_MODE_KANJI2	776 Kanji characters	3390 Kanji characters	Invalid
SCE_HANDWRITING_MODE_HANGUL	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANJA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	Invalid

	SCE_HANDWRITING_DIC_KOREAN Korea (Hangul + Korean Hanja)	SCE_HANDWRITING_DIC_HANGUL Hangul	SCE_HANDWRITING_DIC_GERMAN Germany
SCE_HANDWRITING_MODE_NUMBER	10 numeric characters	10 numeric characters	10 numeric characters
SCE_HANDWRITING_MODE_UALPHA	26 alphabetic uppercase characters	26 alphabetic uppercase characters	26 alphabetic uppercase characters 3 Latin uppercase characters
SCE_HANDWRITING_MODE_LALPHA	26 alphabetic lowercase characters	26 alphabetic lowercase characters	26 alphabetic lowercase characters 4 Latin lowercase characters
SCE_HANDWRITING_MODE_SYMBOL	, . ' " – ! ? 7 characters	, . ' " – ! ? 7 characters	, . ' " – ! ? 7 characters
SCE_HANDWRITING_MODE_HIRA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KATA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_GREEK	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI1	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI2	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANGUL	2350 Hangul characters	2350 Hangul characters	Invalid
SCE_HANDWRITING_MODE_HANJA	4888 Korean Hanja characters	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	Invalid

	SCE_HANDWRITING_DIC_DUTCH Holland	SCE_HANDWRITING_DIC_FRENCH France	SCE_HANDWRITING_DIC_ITALIAN Italy
SCE_HANDWRITING_MODE_NUMBER	10 numeric characters	10 numeric characters	10 numeric characters
SCE_HANDWRITING_MODE_UALPHA	26 alphabetic uppercase characters 12 Latin uppercase characters	26 alphabetic uppercase characters 15 Latin uppercase characters	26 alphabetic uppercase characters 7 Latin uppercase characters
SCE_HANDWRITING_MODE_LALPHA	26 alphabetic lowercase characters 12 Latin lowercase characters	26 alphabetic lowercase characters 15 Latin lowercase characters	26 alphabetic lowercase characters 7 Latin lowercase characters
SCE_HANDWRITING_MODE_SYMBOL	, . ' " – ! ? 7 characters	, . ' " – ! ? 7 characters	, . ' " – ! ? 7 characters
SCE_HANDWRITING_MODE_HIRA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KATA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_GREEK	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI1	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI2	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANGUL	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANJA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	Invalid

	SCE_HANDWRITING_DIC_SPANISH Spain	SCE_HANDWRITING_DIC_PORTUGUESE Portugal	SCE_HANDWRITING_DIC_RUSSIAN Russia
SCE_HANDWRITING_MODE_NUMBER	10 numeric characters	10 numeric characters	10 numeric characters
SCE_HANDWRITING_MODE_UALPHA	26 alphabetic uppercase characters 7 Latin uppercase characters	26 alphabetic uppercase characters 13 Latin uppercase characters	26 alphabetic uppercase characters
SCE_HANDWRITING_MODE_LALPHA	26 alphabetic lowercase characters 7 Latin lowercase characters	26 alphabetic lowercase characters 13 Latin lowercase characters	26 alphabetic lowercase characters
SCE_HANDWRITING_MODE_SYMBOL	, . ' " – ! ? ¡ ¿ 9 characters	, . ' " – ! ? 7 characters	, . ' " – ! ? 7 characters
SCE_HANDWRITING_MODE_HIRA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KATA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_GREEK	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI1	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI2	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANGUL	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANJA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	33 Russian uppercase characters
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	33 Russian lowercase characters

	SCE_HANDWRITING_DIC_DAN_NOR Denmark + Norway	SCE_HANDWRITING_DIC_SWE_FIN Sweden + Finland	SCE_HANDWRITING_DIC_ICELANDIC Iceland
SCE_HANDWRITING_MODE_NUMBER	10 numeric characters	10 numeric characters	10 numeric characters
SCE_HANDWRITING_MODE_UALPHA	26 alphabetic uppercase characters 3 Latin uppercase characters	26 alphabetic uppercase characters 3 Latin uppercase characters	26 alphabetic uppercase characters 10 Latin uppercase characters
SCE_HANDWRITING_MODE_LALPHA	26 alphabetic lowercase characters 3 Latin lowercase characters	26 alphabetic lowercase characters 3 Latin lowercase characters	26 alphabetic lowercase characters 10 Latin lowercase characters
SCE_HANDWRITING_MODE_SYMBOL	, . ' " – ! ? 7 characters	, . ' " – ! ? 7 characters	, . ' " – ! ? 7 characters
SCE_HANDWRITING_MODE_HIRA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KATA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_GREEK	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI1	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI2	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANGUL	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANJA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	33 Russian uppercase characters
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	33 Russian lowercase characters

	SCE_HANDWRITING_DIC_POLISH Poland	SCE_HANDWRITING_DIC_SIMPLIFIED_CHINESE Chinese (Simplified)	SCE_HANDWRITING_DIC_TRADITIONAL_CHINESE Chinese (Traditional)
SCE_HANDWRITING_MODE_NUMBER	10 numeric characters	10 numeric characters	10 numeric characters
SCE_HANDWRITING_MODE_UALPHA	26 alphabetic uppercase characters 9 Latin uppercase characters	26 alphabetic uppercase characters	26 alphabetic uppercase characters
SCE_HANDWRITING_MODE_LALPHA	26 alphabetic lowercase characters 9 Latin lowercase characters	26 alphabetic lowercase characters	26 alphabetic lowercase characters
SCE_HANDWRITING_MODE_SYMBOL	, . ' " – ! ? 7 characters	, . ’ ” – ! ? 。 、 : ; 11 characters	, . ’ ” – ! ? 。 、 : ; 11 characters
SCE_HANDWRITING_MODE_HIRA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KATA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_GREEK	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_KANJI1	Invalid	3755 simplified Chinese characters	5401 traditional Chinese characters *1
SCE_HANDWRITING_MODE_KANJI2	Invalid	3008 simplified Chinese characters	7650 traditional Chinese characters *1
SCE_HANDWRITING_MODE_HANGUL	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HANJA	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_URUSSIAN	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_LRUSSIAN	Invalid	Invalid	Invalid
SCE_HANDWRITING_MODE_HONGKONG	Invalid	Invalid	HKSCS 4510 characters *2

If SCE_HANDWRITING_MODE_ALL is specified, all the characters of all the dictionaries are valid.

*1 There are 2 characters ("兀" and "㪍") registered with the same letter form.
The 2 duplicate Chinese characters return the character code (Unicode UCS-4) of the Chinese characters included in the CJK unified ideographs.
In the Big5 standard, "兀" belongs to the 1st level and the 2nd level, but the "兀" of the 2nd level is not part of the characters targeted for recognition.
Based on the above, the number of characters of the 2nd level of the Big 5 standard is 7652, but the number of characters that is recognized is 7650 characters.

*2 Of the 4511 characters of HKSCS 2004, 4510 characters are characters targeted for recognition.
"龜" U+F907 is not among the characters targeted for recognition.
The 7 Eten extension characters "碁, 銹, 恒, 裏, 墻, 粧, 嫺" are included in these characters.

See Also

`sceHandwritingInit(),sceHandwritingRecognize()`

libhandwriting Functions (Character Registration Function)

sceHandwritingRegisterInit

Initialize character registration function

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingRegisterInit (
        SceUInt32 count,
        ScePVoid regBuffer
);
```

Calling Conditions

Multithread safe.

Arguments

count	Maximum number of registered characters The maximum number of registered characters that can be specified is SCE_HANDWRITING_MAX_REGISTER_COUNT.
regBuffer	Memory addresses to which registered dictionary was loaded. Memory of *count* x SCE_HANDWRITING_REGISTER_BLKSIZE size is required.

Return Values

Returns a negative value for errors.

Value	Hexadecimal	Description
SCE_OK	0x0	Normal termination
SCE_HANDWRITING_ERROR_INVALID_POINTER	0x80105600	Argument *regBuffer* is invalid
SCE_HANDWRITING_ERROR_INVALID_PARAM	0x80105601	Invalid argument value
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function initializes the registered dictionary used for the character registration function. The required memory size of the registered dictionary is *count* x SCE_HANDWRITING_REGISTER_BLKSIZE bytes.

Call this function after allocating the required memory at the application level.

If a new dictionary is to be registered, the memory must be zero-cleared at the application level.

Write (sceHandwritingRegisterSet() and sceHandwritingRegisterDelete()) and read (sceHandwritingRegisterGetResult() and sceHandwritingRegisterInfo()) are executed for the registered dictionaries specified with this function.

Multiple instances of sceHandwritingRegisterInit() cannot be made to work at the same time.

If sceHandwritingRegisterInit() ends normally, do not access the memory area of *regBuffer* until sceHandwritingRegisterTerm() is called.

See Also

sceHandwritingRegisterTerm(),sceHandwritingRegisterSet(),
sceHandwritingRegisterGetResult(),sceHandwritingRegisterDelete(),
sceHandwritingRegisterInfo()

sceHandwritingRegisterTerm

Terminate character registration function

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingRegisterTerm ();
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns a negative value for errors.

Value	Hexadecimal	Description
SCE_OK	0x0	Normal termination
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function is called to terminate the character registration function.

See Also

sceHandwritingRegisterInit()

sceHandwritingRegisterSet

Add character code and its corresponding handwritten character to the registered dictionary

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingRegisterSet (
        SceWChar32 code,
        const SceHandwritingPointsData *data
);
```

Calling Conditions

Multithread safe.

Arguments

code Character code to be registered (Unicode UCS-4)
data Handwriting data to be registered

Return Values

Returns the number of registrations upon normal termination. Returns a negative value for errors.

Value	Hexadecimal	Description
Positive number	-	Number of registrations (normal termination)
SCE_HANDWRITING_ERROR_INVALID_POINTER	0x80105600	Invalid argument pointer
SCE_HANDWRITING_ERROR_INVALID_PARAM	0x80105601	Invalid argument value
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_DIC_FULL	0x80105607	No more items can be registered
SCE_HANDWRITING_ERROR_CODE_ERROR	0x80105608	Invalid character code
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function adds the specified writing data and character codes (Unicode UCS-4) to the registered dictionary.

Before using this function, the registered dictionaries, to which characters are to be added, must be initialized with `sceHandwritingRegisterInit()`.

The character codes that can be registered are 0x00000020 to 0x0000E7FF, 0x0000F900 to 0x0000FFFD, JIS 3rd level, JIS 4th level, and 4-byte character codes used in HKSCS.

It is recommended not to use the control code part (*).

To register original characters, use the range of 0x0000E000 to 0x0000E7FF.

The maximum number of items that can be registered is the number of characters specified with `sceHandwritingRegisterInit()`.

* Control code part 0x00000000 to 0x0000001F, 0x0000007F
0x00002000 to 0x0000200f, 0x00002011
0x00002028 to 0x0000202f, 0x0000205f to 0x0000206f
0x0000fe00 to 0x0000feff, 0x0000fff0 to 0x0000ffff

See Also

`SceHandwritingPointsData`, `sceHandwritingRegisterInit()`

sceHandwritingRegisterGetResult

Get result of character registration function

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingRegisterGetResult (
        SceHandwritingRegisterData *data
);
```

Calling Conditions

Multithread safe.

Arguments

data Character code array structure that stores the result

Return Values

Returns the number of results upon normal termination. Returns a negative value for errors.

Value	Hexadecimal	Description
Value of 0 or greater	-	Number of results (normal termination)
SCE_HANDWRITING_ERROR_INVALID_POINTER	0x80105600	Invalid argument pointer
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function gets the result in the registered dictionary only. Before using this function, the sceHandwritingRecognize() function must have terminated normally. The list of character codes is stored to the *codes* member of the SceHandwritingRegisterData structure. The storing order is from the highest recognition assessment.

The number of candidates that can be obtained is up to the number specified by SCE_HANDWRITING_MAX_REGISTER_CANDIDATES.

See Also

SceHandwritingRegisterData, sceHandwritingRecognize()

sceHandwritingRegisterDelete

Delete registered character from the registered dictionary

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingRegisterDelete (
        SceWChar32 code
);
```

Calling Conditions

Multithread safe.

Arguments

code Character code to be deleted (Unicode UCS-4)

Return Values

Returns the number of registered items after deletion upon normal termination. Returns a negative value for errors.

Value	Hexadecimal	Description
Value of 0 or greater	-	Number of registered items (normal termination)
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_CODE_ERROR	0x80105608	Invalid character code (the specified code was not found)
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function deletes the specified character code (Unicode UCS-4) from the registered dictionary. If multiple specified character codes (Unicode UCS-4) have been registered, the function performs multiple deletions.

Before using this function, the registered dictionaries, from which characters are to be deleted, must be initialized with `sceHandwritingRegisterInit()`.

See Also

`sceHandwritingRegisterInit()`

sceHandwritingRegisterInfo

Get registration information of character registration function

Definition

```
#include <libhandwriting/libhandwriting_api.h>
SceInt32 sceHandwritingRegisterInfo (
        SceWChar32 *codes
);
```

Calling Conditions

Multithread safe.

Arguments

codes Pointer to character code (Unicode UCS-4) array for storing the obtained result

Return Values

Returns the number of registered items upon normal termination. Returns a negative value for errors.

Value	Hexadecimal	Description
Positive number	-	Number of registered items (normal termination)
SCE_HANDWRITING_ERROR_INVALID_POINTER	0x80105600	Invalid argument pointer
SCE_HANDWRITING_ERROR_NOT_INITIALIZED	0x80105604	Initialization has not been performed
SCE_HANDWRITING_ERROR_INTERNAL	0x80105610	Internal error

Description

This function gets the list of character codes (Unicode UCS-4) currently registered to the registered dictionary. Since the maximum number of items that is obtained being equivalent to the *count* argument of the sceHandwritingRegisterInit() function, a buffer of *count* x sizeof(SceWChar32) bytes is required.

Before using this function, the registered dictionary must be set with sceHandwritingRegisterInit().

See Also

sceHandwritingRegisterInit()