

KCode Reference Manual

1. Introduction

This document contains information on programming with KCode (Korean Code) Conversion APIs. Section 2 describes the information on KCode API Solution One and Section 3 describes the information on KCode API Solution Two.

Solution One and Solution Two differ in the way they convert the Korean Hangul Code (Unicode Chart Name [Hangul Syllables]: Code Range U+AC00 ~ U+D7A3).

1.1. Introduction to KCode Solution One and Solution Two

? Conversion Solution One (KCSO hereafter)

- KCSO converts Unicode Chart Name [Hangul Syllables] only to KS C 5601 1987 Hangul Syllables, and vice versa. KS C 5601 1987 contains only 2,350 Hangul Syllables, though there are 11,172 Korean Hangul Syllables. So, in case that a Unicode Hangul Syllable cannot be mapped to any KS C 5601 Hangul Syllable, KCSO converts one syllable to two syllables. For more detailed information, refer to the [KCode Technical Document].
- When convert Unicode Hangul Syllable to KS C 5601 Hangul Syllable, KCSO first decomposes a syllable to CHOSUNG + JUNGSUNG + JONGSUNG, and then converts the CHOSUNG, JUNGSUNG, JONGSUNG to Korean Combination Code, respectively. Finally KCSO combines the Korean Combine Code character encoding syllable to KS C 5601 syllable.
- CHOSUNG, JUNGSUNG, JONGSUNG are approximately equivalent to onset, peak, coda of a syllable structure. (? Refer to the [KCode Technical Document] for further information).
- Korean Code System can be divided into two major encoding methods: First, the character encoding method, and second, the syllable encoding method. You can decompose any syllable to CHOSUNG, JUNGSUNG, JONGSUNG, if you use character encoding method, for example Korean Combination Code, Unicode [Hangul Syllables] (version 2.0 or later). But if you use syllable encoding method, you cannot extract the CHOSUNG, JUNGSUNG, JONGSUNG values from the syllable directly, for example, KS C 5601, Microsoft Unified Hangul. (? Refer to the [KCode Technical Document] for further information).
- In summary, KCSO can only convert 2,350 characters

? Conversion Solution Two (KCST hereafter)

- KSCT can convert Unicode Chart Name [Hangul Syllables] to KS C 5601 and Microsoft Unified Hangul, and vice versa. And more, KSCT can be extended to convert Korean Codes among the Unicode [Hangul Syllables], KS C 5601, Korean Commercial Combination Code Systems.
- In case that Unicode [Hangul Syllables] are converted to/from KS C 5601, KCST works the same as KCSO, but uses different conversion method.
- When converting the Unicode [Hangul Syllables] to/from Microsoft Unified Hangul conversion, you can process all of the 11,172 Korean Hangul Syllables.

2. KCSO API Function Reference

? What can KCSO do and cannot do.

- KCSO converts Unicode String to KS C 5601 String, so KCSO can convert not only Unicode [Hangul Syllables], but also any other Unicode Characters contained in the Source String only if the Unicode Character has the counterpart in KS C 5601, otherwise the Unicode Character is mapped into a default character (Currently ASCII Space: 0x20 ? See [KCode Technical Document] for further Information).
- Unicode [Hangul Syllables] contains 11,172 Hangul Syllables, so KCSO extends Hangul Syllable, which is not contained in KS C 5601, and maps that syllable to two syllables: that is, Unicode 2-byte character (one syllable) is mapped into KS C 5601 4-byte characters (two syllables).

2.1. KCSO API Functions

The following functions can be called by any programs which use the KCSO to convert Unicode String to/from KS C 5601 String.

2.1.1. Unicode to KS C 5601 String Conversion

```
int Uni2KSCString( wchar_t *uniString, unsigned char *kscString)
```

This function converts the Unicode String to KS C 5601 String. The return value and the function arguments are as follows:

? wchar_t *uniString

- A pointer to the wide character 'wchar_t' type array string or dynamically allocated string buffer.
- This argument points to the Unicode encoded String that will be converted to KS C 5601 encoded String.
- Also, Unicode String 'uniString' must end with NULL character, i.e., 2-byte 0x0000.

? unsigned char *kscString

- This argument points to the 'unsigned char' type string buffer. Before this parameter is used, the memory space must be allocated to this pointer to contain the converted

string. So, before call this function, you must allocate appropriate memory space to this pointer, and then call this function.

- 'unsigned char *kscString' holds the KS C 5601 encoded string that is converted from Unicode encoded string, and ends with string ending character, i.e., 1-byte 0x00, or '\0' character.

? Return value:

- This function returns the length (number of bytes) of 'kscString'. The return value may be less or greater than the length of Unicode encoded 'uniString'.
- If the 'uniString' contains only ASCII Characters (Unicode Code Range: U+0x0020 ~ U+0x007E), then the length of the 'uniString' (i.e., wcslen(uniString)) is the same as the length of 'kscString' (i.e., strlen(kscString)).

? For more information about how the string length varies, See the [KCode Technical Document].

```
int KSC2UniString(unsigned char *kscString, wchar_t *uniString)
```

This function converts the KS C 5601 String to Unicode String. The return value and the function arguments are as follows:

? unsigned char *kscString

- A pointer to the 'unsigned character' type array string or dynamically allocated string buffer.
- This argument points to the KS C 5601 encoded String that will be converted to Unicode encoded String.
- Also, KS C 5601 encoded 'kscString' must end with NULL character, i.e., 1-byte 0x00 or '\0' Character.

? wchar_t *uniString

- This argument points to the Wide Character 'wchar_t' type string buffer. Before the parameter is used, the memory space must be allocated to this pointer to contain the converted string. So, before call this function, you must allocate appropriate memory space to this pointer, and then call this function.
- Wide Character type pointer 'wchar_t *uniString' holds the Unicode encoded string that is converted from KS C 5601 encoded string, and ends with string ending character, i.e., 2-byte 0x0000.

? Return value

- This function returns the length of 'uniString'. The return value may be equivalent to or greater than the length of KS C 5601 encoded 'kscString'.
- For more information about how the string length varies, See the [KCode Technical Document].

```
int KSCStrLenOfUni(wchar_t *uStr)
```

This function calculates the number of bytes to store the corresponding KS C 5601 encoded String.

? wchar_t *uStr

- Unicode encoded string that will be converted to KS C 5601 String.

? Return value

- This function returns the number of bytes that must be allocated to KS C 5601 String 'unsigned char *kscStr', which will contain the converted string.
- In KS C 5601 Code System, Hangul Syllables & other Graphic Characters take 2-byte space. So one Unicode character can be converted to 1-byte character or 2-byte character string (See the [KCode Technical Document] for further Information).

- The value this function returns are not exactly the same as the memory space that is needed to convert Unicode String to KS C 5601 String, but the return value is equivalent to or greater than the minimum memory space to be used.

```
int UniLenOfKSCStr(char *kStr)
```

This function calculates the number of Wide Characters to store the corresponding KS C 5601 encoded String.

? char *kStr

- KS C 5601 encoded string that will be converted to Unicode encoded string.

? Return value

- This function returns the number of wide character(s) that is needed to hold the converted Unicode String.
- The return value may be less than or equivalent to the length of 'kstr'.

```
INT32 KSC2Unicode_calcLen(BYTE *str,
                           BOOL isNullTerminated,
                           UINT32 readBytes,
                           UINT32 *strByteLen)
```

This function calculates the length of a KS C 5601 encoded string. In Korea, one character may take 1-byte or 2-bytes. This is because that ASCII 1-byte characters and KS C 5601 2-bytes characters can be used mixed in a text file, and they are discriminated with the MSB(Most Significant Bit) 1-Bit Flag. Therefore, PCS Phones also use 1-byte characters and 2-bytes characters mixed in one code system.

? BYTE *str

- The pointer to the string of which the length will be calculated.

? BOOL isNullTerminated

- If this flag is TRUE, the end of string is ZERO, i.e. 0x00 or '\0', and no ZERO character is inserted in the middle of the 'BYTE *str', otherwise, the 'BYTE *str' argument can contain ZERO byte in the middle of the 'BYTE *str'.

? UINT32 readBytes

- The actual length of bytes 'BYTE *str' holds.

? UINT32 *strByteLen

- This pointer can hold different length according the flag value of 'BOOL isNullTerminated'.
- If 'BOOL isNullTerminated' is TRUE then the 'readBytes' is the same as the value this pointer will hold, otherwise, this pointer holds the length from the start to the position where the ZERO byte ('\0' or 0x00) occurs first.

3. KCST API Functions

KCode Solution Two(2) can be used with the same method as KCSO API Functions. Also, KCST has added functions regarding to the Microsoft Unified Hangul to/from Unicode String Conversion.

In KCST, the following two functions are added.

```
int UnifiedHan2UniString(unsigned char *uhString, wchar_t *uniString)
```

This function has the same interface with the function ‘int KSC2UniString(unsigned char *kscString, wchar_t *uniString)’ which is defined in KCSO and KCST. But, unlike the ‘KSC2UniString’ function, this function can convert not only the KS C 5601 2,350 Hangul Syllables, but also Microsoft Unified Hangul 11,172 Hangul Syllables (KS C 5601 Hangul Syllables are included in Microsoft Unified Hangul Syllables).

```
int Uni2UnifiedHanString(wchar_t *uniString, unsigned char *uhString)
```

This function has the same interface with the function ‘int Uni2KSCString(wchar_t *uniString, unsigned char *kscString)’ which is defined in KCSO and KCST. But, unlike the ‘Uni2KSCString’, all of the 11,172 Unicode [Hangul] Syllables can be converted to the Microsoft Unified Hangul.

In case of KS C 5601, some Unicode Hangul Syllables, which cannot be mapped to KS C 5601 Syllables, are expanded to two KS C 5601 Syllables (See the [KCode Technical Document] for further information).

```
int extKSCStrLenOfUni(wchar_t *uStr)  
extKSCStrLenOfUni(wchar_t *uStr)
```

This function works the same way as the function ‘int KSCStrLenOfUni(wchar_t *uStr)’ except that the source string ‘uStr’ should be converted to Microsoft Unified Hangul String.

```
int UniLenOfextKSCStr(char *kStr)  
int LenOfextKSCStr(char *kStr)
```

except that the source string ‘kStr’ should be Microsoft Unified Hangul String.

```
INT32 extKSC2Unicode_calcLen(  BYTE *str,  
                                BOOL isNullTerminated,  
                                UINT32 readBytes,  
                                UINT32 *strByteLen)
```

The following functions are used with the same way as defined in KCSO.

```
int KSC2UniString(unsigned char *kscString, wchar_t *uniString)
```

```
int Uni2KSCString(wchar_t *uniString, unsigned char *kscString)
```

```
int KSCStrLenOfUni(wchar_t *uStr)
```

```
int UniLenOfKSCStr(char *kStr)
```

```
INT32 KSC2Unicode_calcLen(BYTE *str,  
                           BOOL isNullTerminated,  
                           UINT32 readBytes,  
                           UINT32 *strByteLen)
```