Demo Project for String Operation Functions

Table of Contents

1 Overview and Operation

[Overview]
This demo project shows how to use MACRO String Operation Functions, the syntax and descriptions are listed below:

Syntax	StringGet(read_data[start], device_name, device_type, address_offset, data_count)
Description	Receives data from PLC.
Syntax	StringSet(send_data[start], device_name, device_type, address_offset, data_count)
Description	Sends data to PLC.
Syntax	success = StringCopy (“source”, destination[start]) or success = StringCopy (source[start], destination[start])
Description	Copies one string to another.
Syntax	success = StringDecAsc2Bin(source[start], destination) or success = StringDecAsc2Bin(“source”, destination)
Description	This function converts a decimal string to an integer.
Syntax	success = StringBin2DecAsc (source, destination[start])
Description	This function converts an integer to a decimal string.
Syntax	success = StringDecAsc2Float (source[start], destination) or success = StringDecAsc2Float (“source”, destination)
Description	This function converts a decimal string to floats.
Syntax	success = StringFloat2DecAsc(source, destination[start])
Description	This function converts a float to a decimal string.
Syntax	success = StringHexAsc2Bin (source[start], destination) or success = StringHexAsc2Bin (“source“, destination)
Description	This function converts a hexadecimal string to binary data.
Syntax	success = StringBin2HexAsc (source, destination[start])
Description	This function converts binary data to a hexadecimal string.
Syntax	success = StringMid (source[start], count, destination[start]) or success = StringMid (“string“, start, count, destination[start])

Description	Retrieves a character sequence from the specified offset of the source string and stores it in the destination buffer.
Syntax	length = StringLength (source[start]) or length = StringLength (“source“)
Description	Obtains the length of a string.
Syntax	success = StringCat (source[start], destination[start]) or success = StringCat (“source”, destination[start])
Description	This function appends source string to destination string.
Syntax	ret = StringCompare (str1[start], str2[start]) ret = StringCompare (“string1“, str2[start]) ret = StringCompare (str1[start], “string2”) ret = StringCompare (“string1”, “string2”)
Description	Do a case-sensitive comparison of two strings.
Syntax	ret = StringCompareNoCase(str1[start], str2[start]) ret = StringCompareNoCase(“string1“, str2[start]) ret = StringCompareNoCase(str1[start], “string2”) ret = StringCompareNoCase(“string1”, “string2”)
Description	Do a case-insensitive comparison of two strings.
Syntax	position = StringFind (source[start], target[start]) position = StringFind (“source“, target[start]) position = StringFind (source[start], “target”) position = StringFind (“source“, “target“)
Description	Returns the position of the first occurrence of target string in the source string.
Syntax	position = StringReverseFind (source[start], target[start]) position = StringReverseFind (“source“, target[start]) position = StringReverseFind (source[start], “target”) position = StringReverseFind (“source“, “target“)
Description	Returns the position of the last occurrence of target string in the source string.
Syntax	position = StringFindOneOf (source[start], target[start]) position = StringFindOneOf (“source“, target[start]) position = StringFindOneOf (source[start], “target”) position = StringFindOneOf (“source“, “target“)
Description	Returns the position of the first character in the source

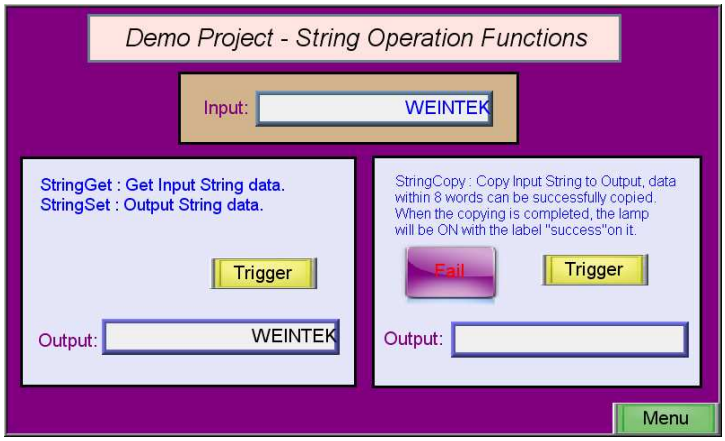
	string that matches any character contained in the target string.
Syntax	success = StringIncluding (source[start], set[start], destination[start]) success = StringIncluding (“source“, set[start], destination[start]) success = StringIncluding (source[start], “set“, destination[start]) success = StringIncluding (“source“, “set“, destination[start])
Description	Retrieves a substring of the source string that contains characters in the set string, beginning with the first character in the source string and ending when a character is found in the source string that is not in the target string.
Syntax	success = StringExcluding (source[start], set[start], destination[start]) success = StringExcluding (“source“, set[start], destination[start]) success = StringExcluding (source[start], “set“, destination[start]) success= StringExcluding (“source“, “set“, destination[start])
Description	Retrieves a substring of the source string that contains characters that are not in the set string, beginning with the first character in the source string and ending when a character is found in the source string that is also in the target string.
Syntax	success = StringToUpper (source[start], destination[start]) success = StringToUpper ("source", destination[start])
Description	Converts all the characters in the source string to uppercase characters and stores the result in the destination buffer.
Syntax	success = StringToLower (source[start], destination[start]) success = StringToLower ("source", destination[start])
Description	Converts all the characters in the source string to lowercase characters and stores the result in the destination buffer.
Syntax	success = StringToReverse (source[start], destination[start]) success = StringToReverse ("source", destination[start])
Description	Reverses the characters in the source string and stores it in the destination buffer.
Syntax	success = StringTrimLeft (source[start], set[start],

	destination[start]) success = StringTrimLeft (“source“, set[start], destination[start]) success = StringTrimLeft (source[start], “set“, destination[start]) success = StringTrimLeft (“source“, “set“, destination[start])
Description	Trims the leading specified characters in the set buffer from the source string.
Syntax	success = StringTrimRight (source[start], set[start], destination[start]) success = StringTrimRight (“source“, set[start], destination[start]) success = StringTrimRight (source[start], “set“, destination[start]) success = StringTrimRight (“source“, “set“, destination[start])
Description	Trims the trailing specified characters in the set buffer from the source string.
Syntax	success = StringInsert (pos, insert[start], destination[start]) success = StringInsert (pos, “insert“, destination[start]) success = StringInsert (pos, insert[start], length, destination[start]) success = StringInsert (pos, “insert“, length, destination[start])
Description	Inserts a string in a specific location within the destination string content.

[Operation]

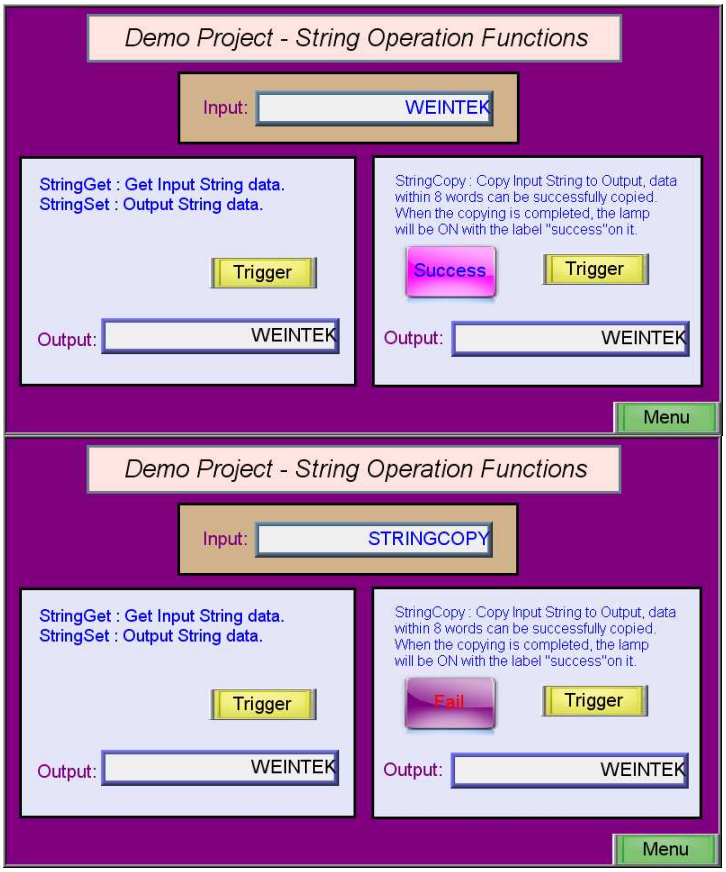
StringGet & StringSet

Input the string, use StringGet to obtain the input string, and then use StringSet to output string data.



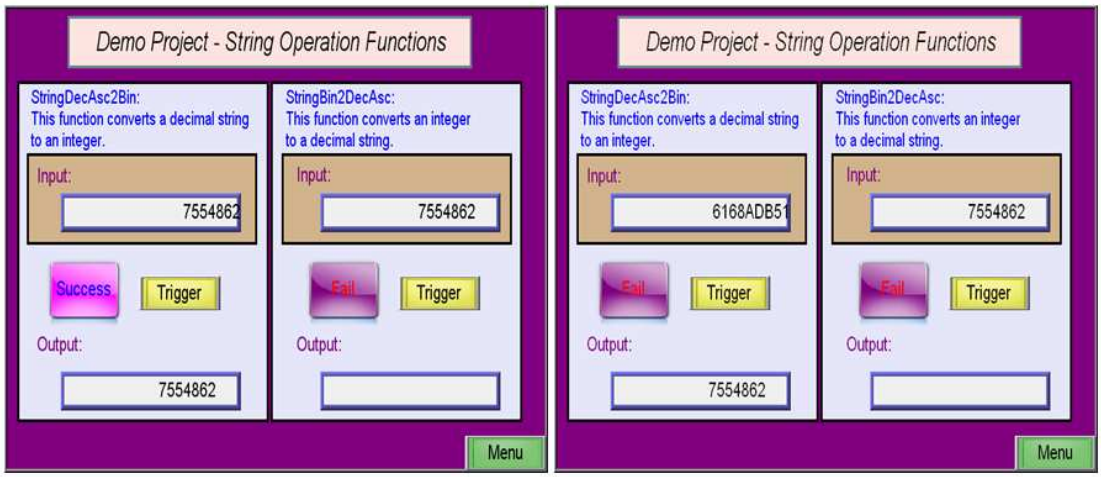
StringCopy

Input the string; use StringCopy to copy the input string then output string data. The target buffer is set to store 8 characters in this demonstration. When input string is within 8 characters it can be successfully copied and an indicator will be shown for notifying. If the copying fails, it shows “Fail”.



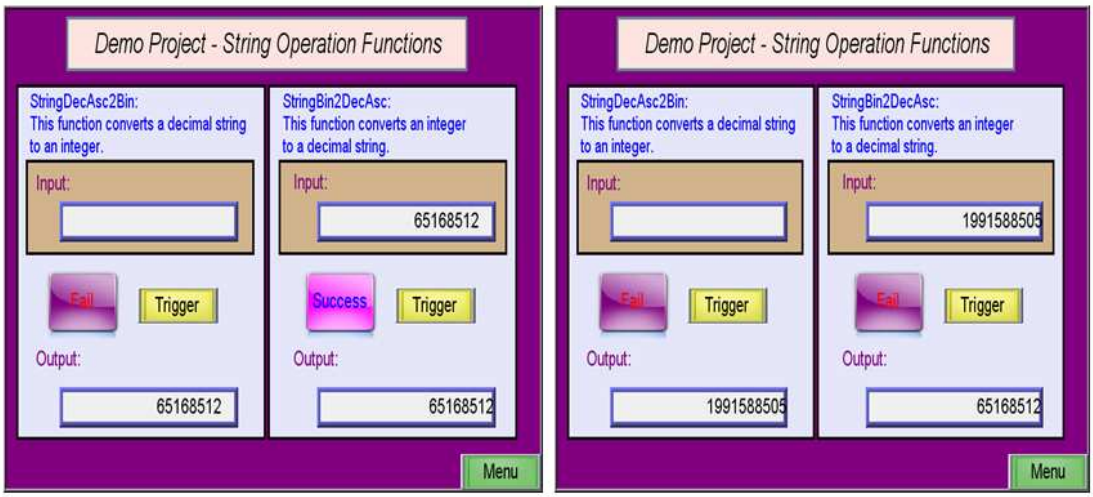
StringDecAsc2Bin

Input a decimal string, use StringDecAsc2Bin to convert it to integer, a “Success” indicator should be shown after conversion. If input characters other then “0~9”, “Fail” indicator will be shown.



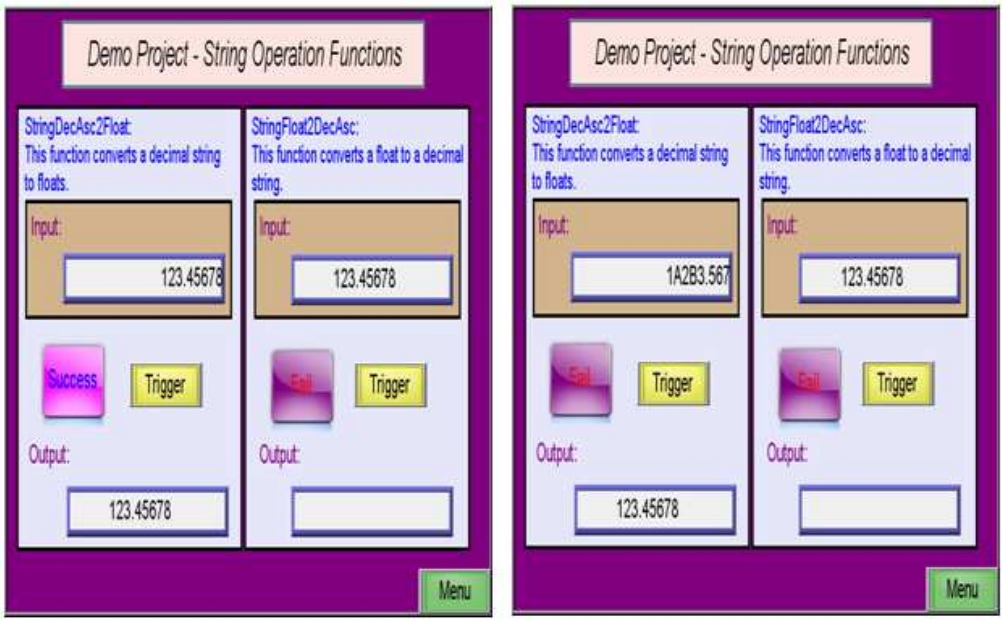
StringBin2DecAsc

Use StringBin2DecAsc to convert integer to decimal string. The target buffer is set to store 8 characters in this demonstration, the conversion can be done within 8 characters, the indicator will show if the conversion “Success” or “Fail”.



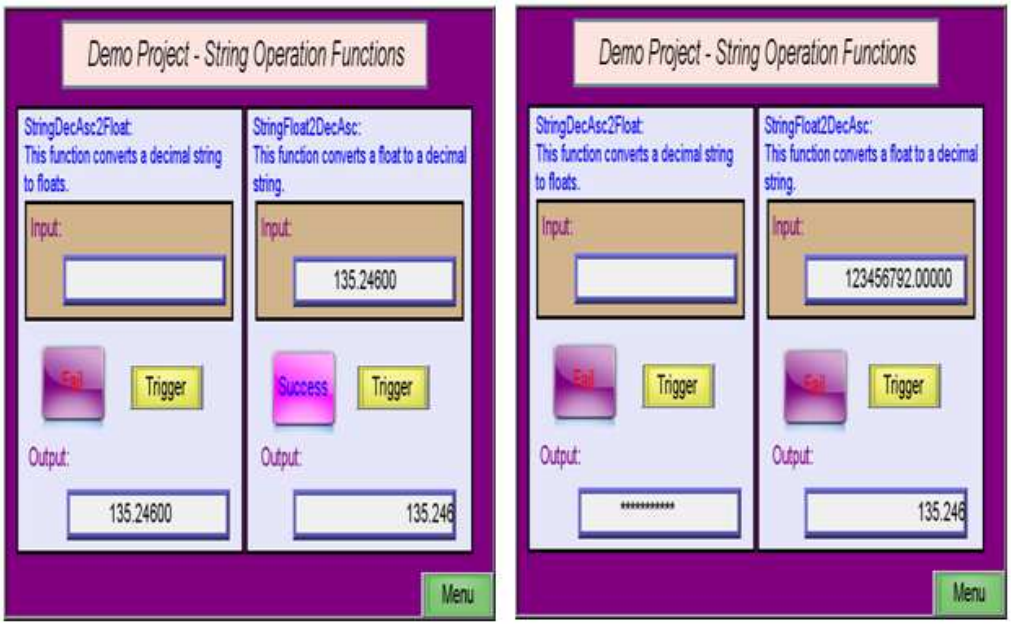
StringDecAsc2Float

Input a decimal string; use StringDecAsc2Float to convert the input string to float. “Success” indicator should be shown after conversion. If input characters other then “0~9”, or “.” , the “Fail” indicator will be shown.



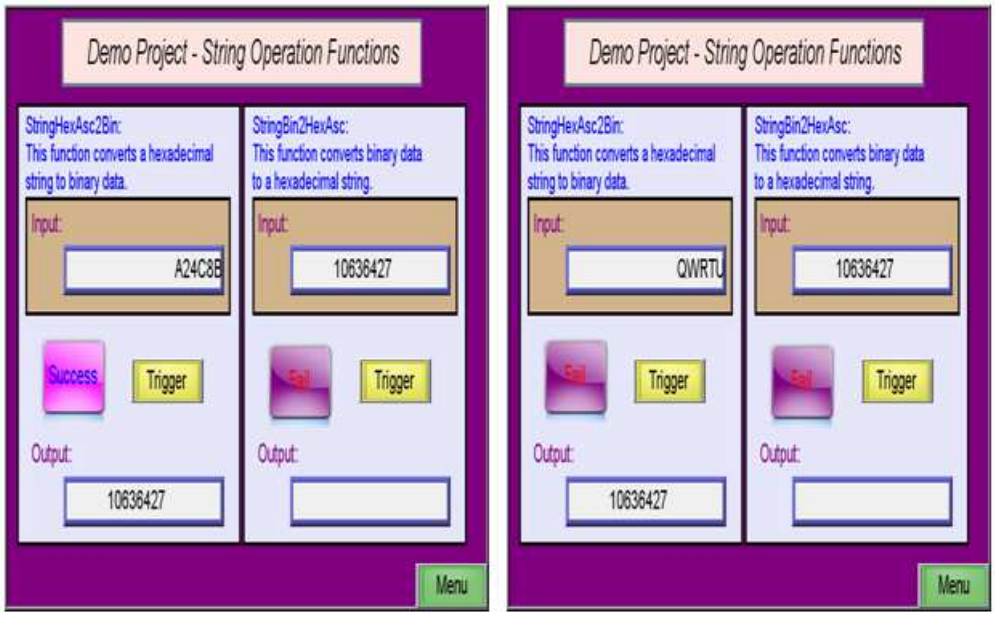
StringFloat2DecAsc

Use StringFloat2DecAsc to convert float to decimal string. The target buffer is set to store 8 characters in this demonstration, the conversion can be done within 8 characters, the indicator will show if the conversion “Success” or “Fail”.



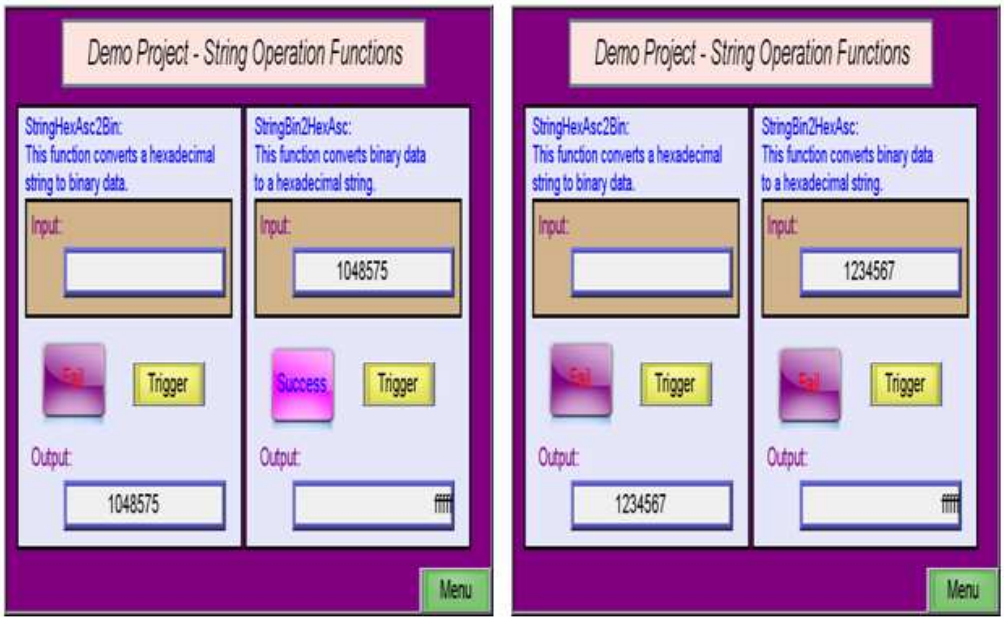
StringHexAsc2Bin

Input a hexadecimal string; use StringHexAsc2Bin to convert the input string to integer. “Success” indicator should be shown after conversion. If input characters other then “0~9”,”a~f” or “A~F”, the “Fail” indicator will be shown.



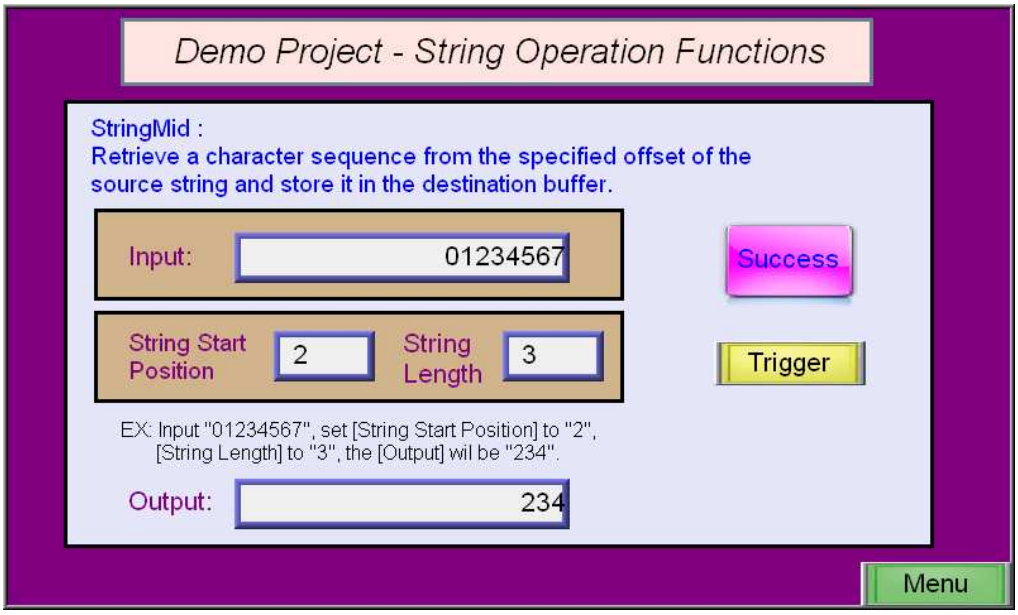
StringBin2HexAsc

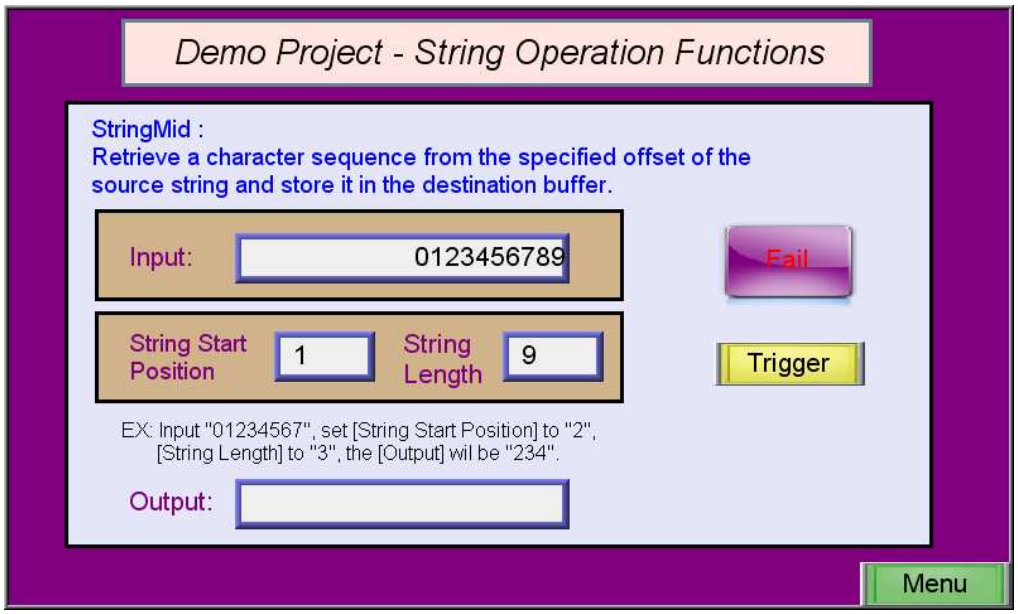
Use StringBin2HexAsc to convert integer to hexadecimal string. The target buffer is set to store 5 characters in this demonstration, the conversion can be done within value 1048575(FFFFF), the indicator will show if the conversion “Success” or “Fail”.



StringMid

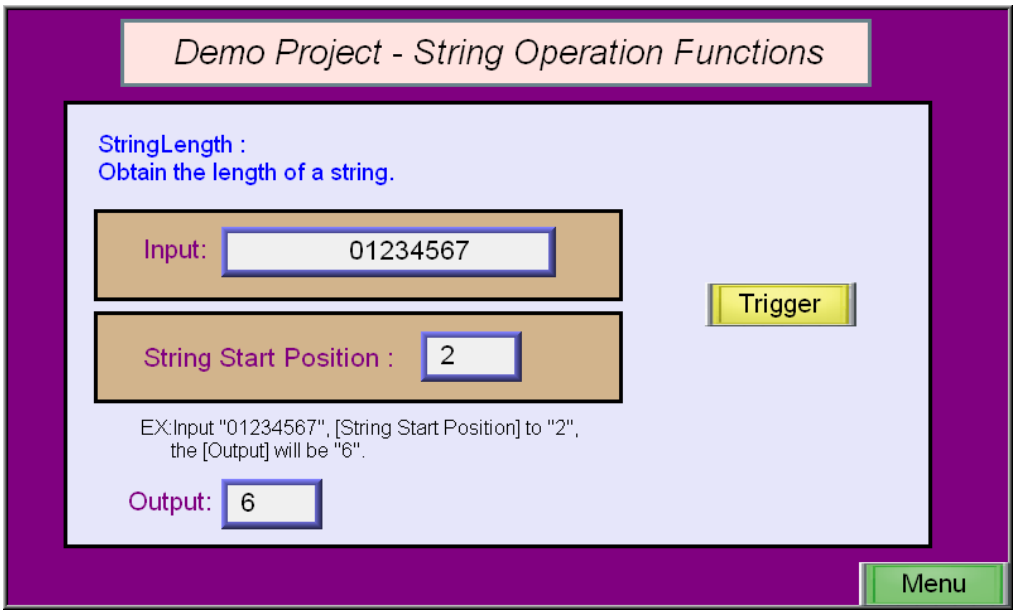
Use StringMid to retrieve a character sequence from the specified offset of the source string. The target buffer is set to store 8 characters in this demonstration, the retrieving can be done within 8 characters, the indicator will show if the conversion “Success” or “Fail”.





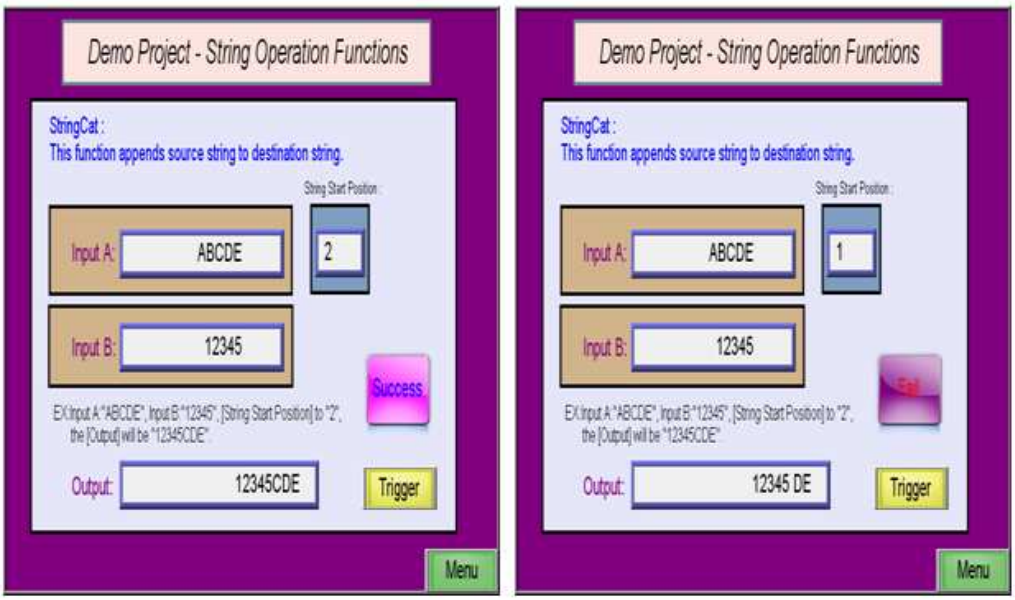
StringLength

Use StringLength to obtain the length of a string, counts from the specified [String Start Position] to the end of the input string.



StringCat

Use StringCat to retrieve a character sequence from [Input A] string and append it to [Input B] string. The target buffer is set to store 8 characters in this demonstration. After appending if the result is within 8 characters, “Success” indicator will be shown.



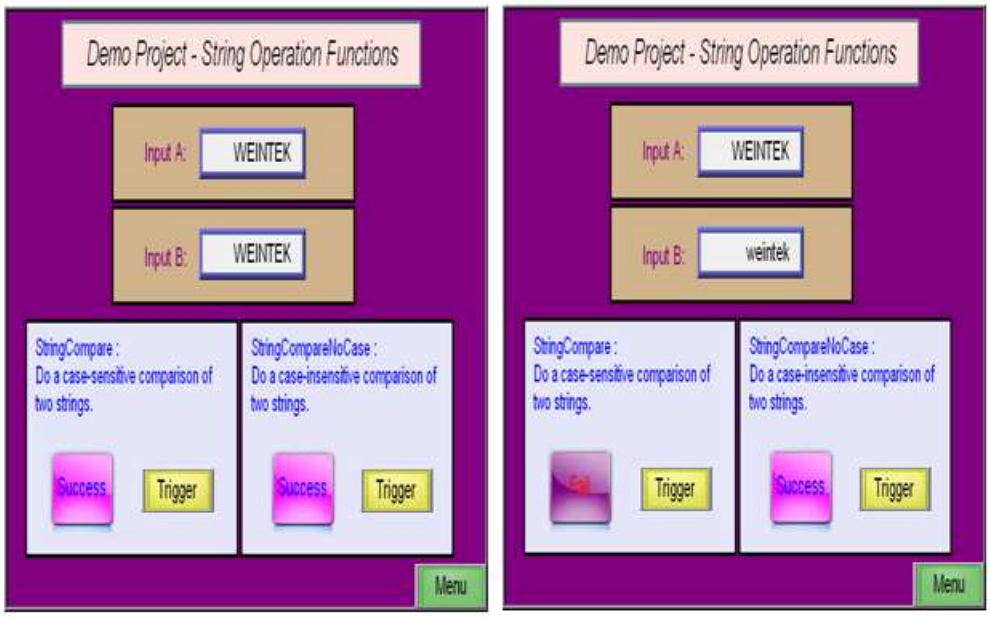
StringCompare

Compare if the contents of [Input A] string & [Input B] string are same, this comparison is case-sensitive.

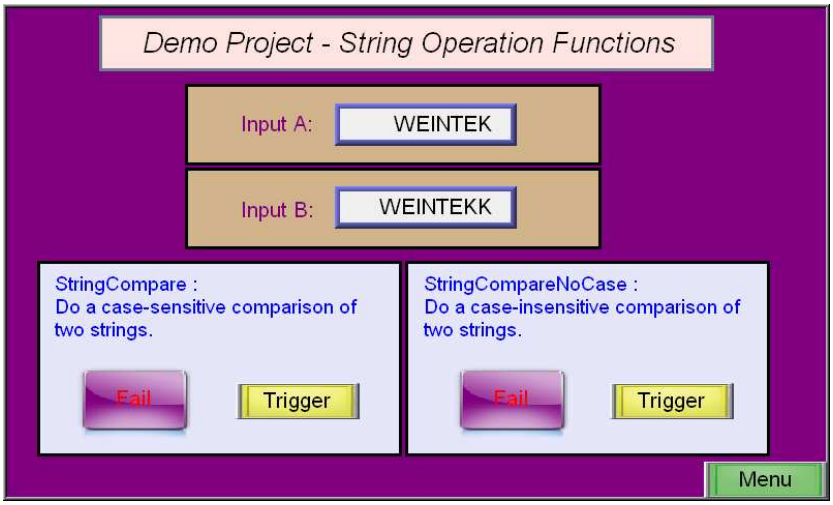
StringCompareNoCase

Compare if the contents of [Input A] string & [Input B] string are same, this comparison is **not** case-sensitive.

Input the exactly same string on [Input A] and [Input B], and trigger the 2 MACRO functions above, “Success” indicator will be shown for both.
Input the same string with different case, “Fail” indicator will be shown for StringCompare function.

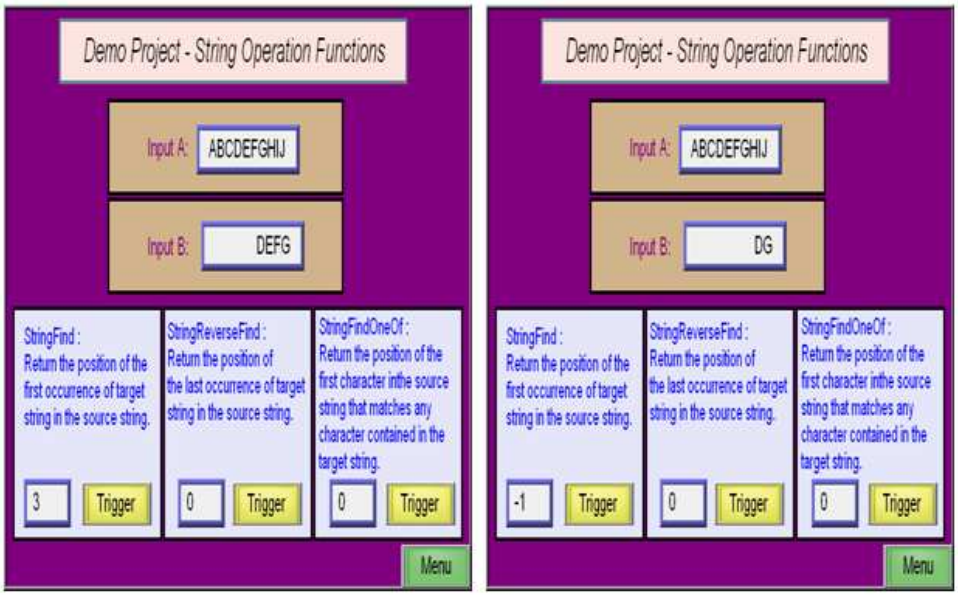


Input different strings and execute these two functions, “Fail” indicator will be shown for both.



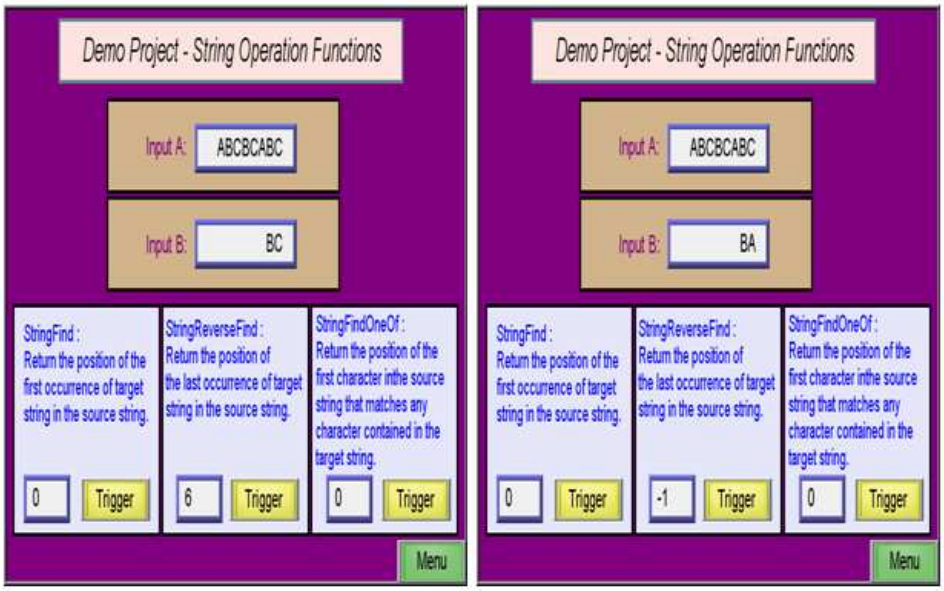
StringFind

Use StringFind to look for the position where [Input B] string (sub) first shows up in [Input A] string (source). The function will return an index value of the start of the substring. The substring must be same as a part of source string, otherwise the function will return “-1”.



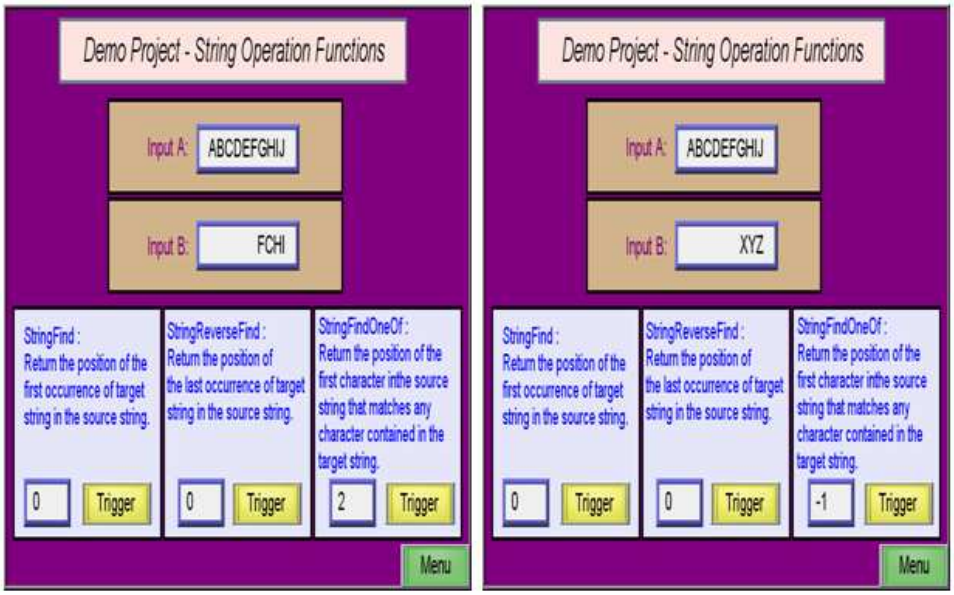
StringReverseFind

Use StringReverseFind to look for the position where [Input B] string (sub) shows up in [Input A] string (source) the last time. The function will return an index value of the start of the substring. The substring must be same as a part of source string, otherwise the function will return “-1”.



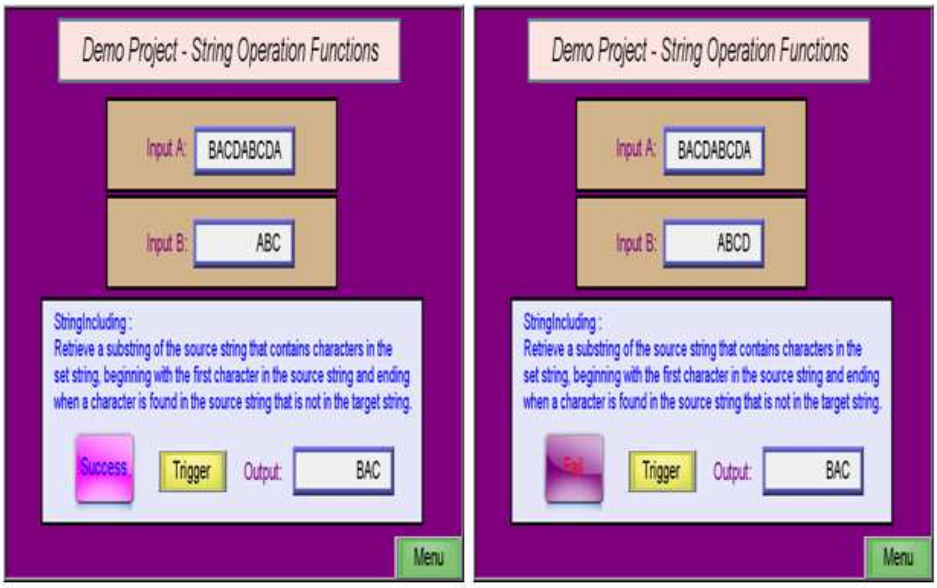
StringFindOneOf

Use StringFindOneOf to search for the position where the character in [Input A] string firstly matches any character in [Input B] string. The function will return the index value of the particular character, if not found, the function will return “-1”.



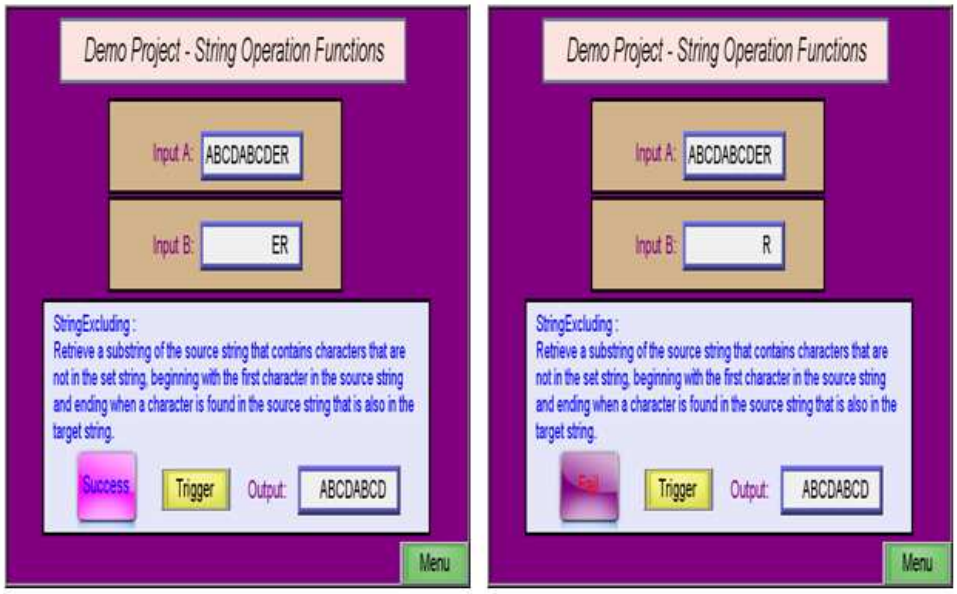
StringIncluding

Use StringFindOneOf to retrieve a substring from [Input A] string that contains characters exist in [Input B] string until encountering a character that can’t be found in [Input B] string. If the retrieved substring exceeds the set 8 characters for buffer to store, “Fail” indicator will be shown.



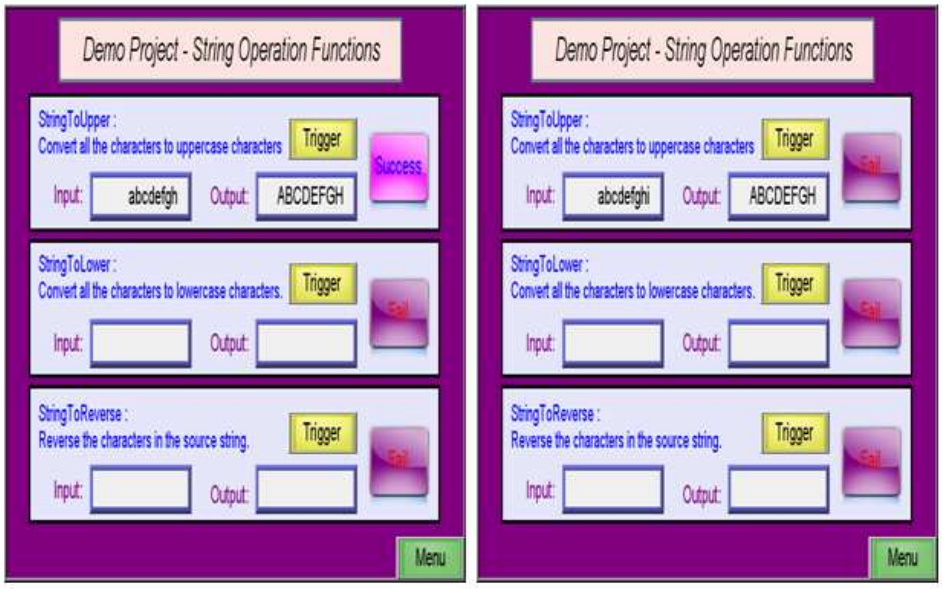
StringExcluding

Use StringFindOneOf to retrieve a substring from [Input A] string that contains characters don't exist in [Input B] string until encountering a character that can be found in [Input B] string. If the retrieved substring exceeds the set 8 characters for buffer to store, "Fail" indicator will be shown.



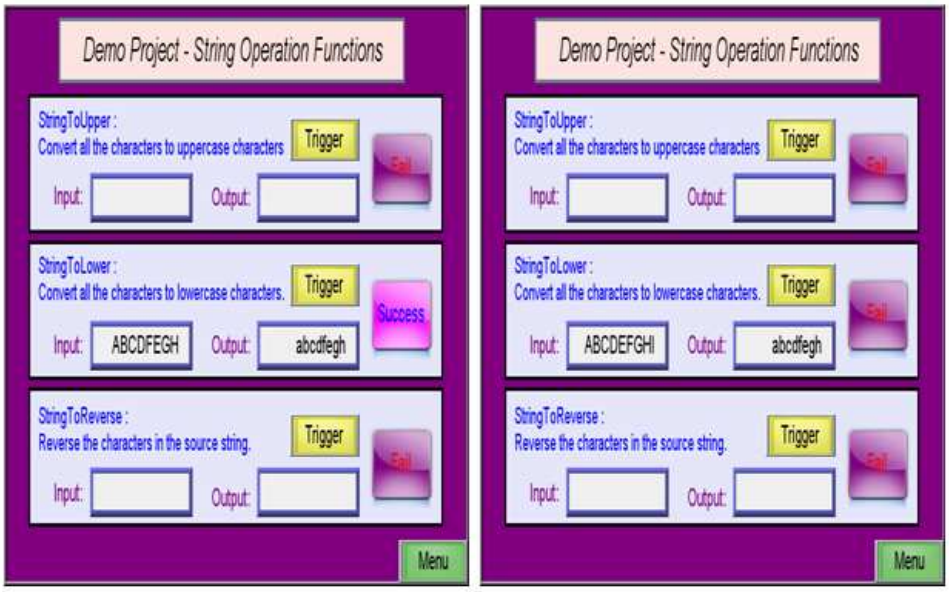
StringToUpper

Use StringToUpper to change all the characters in string to upper case then output. If exceeds the set number of characters, in this demonstration it's 8, "Fail" indicator will be shown.



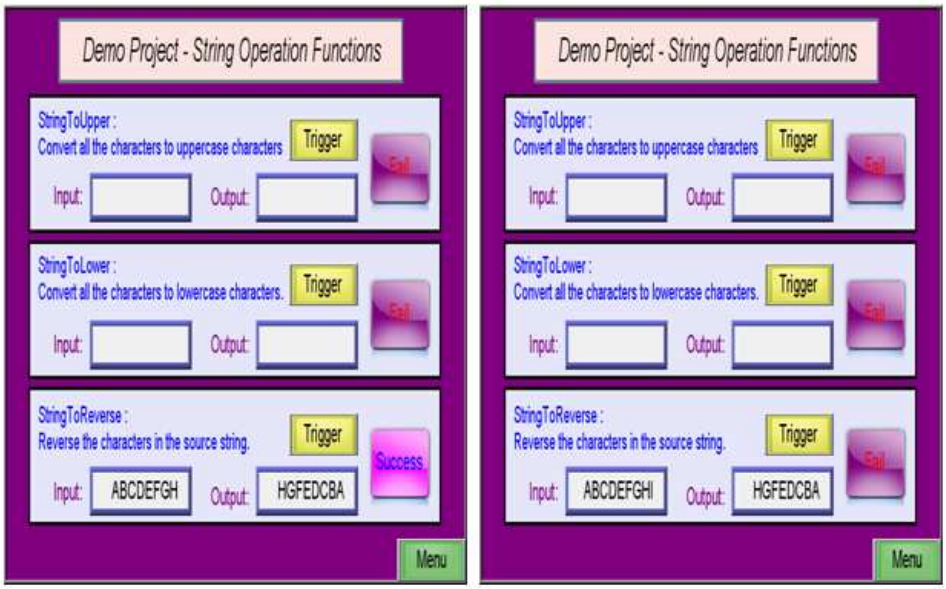
StringToLower

Use StringToLower to change all the characters in string to lower case then output. If exceeds the set number of characters, in this demonstration it's 8, "Fail" indicator will be shown.



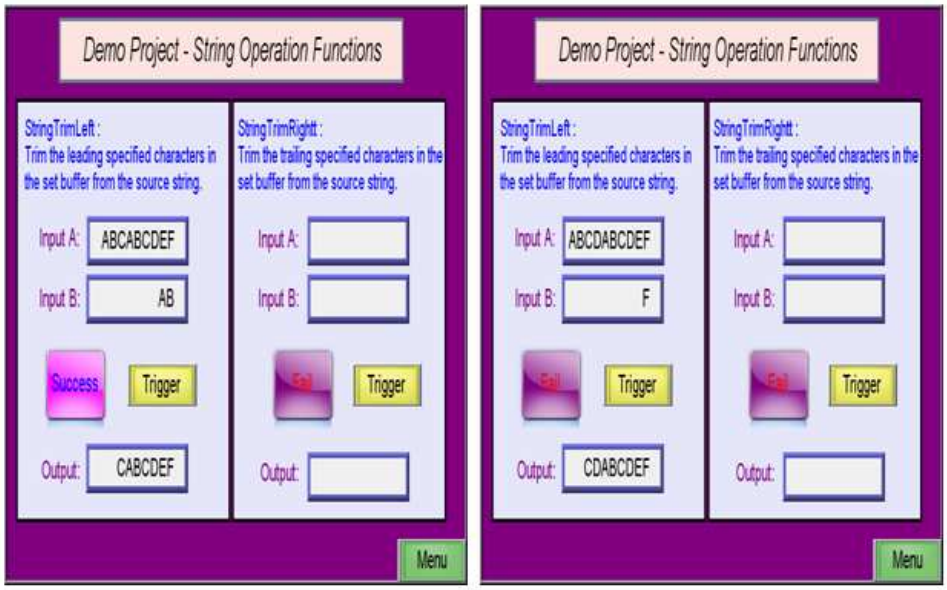
StringToReverse

Use StringToReverse to reverse the input string then output the result. If exceeds the set number of characters, in this demonstration it's 8, "Fail" indicator will be shown.



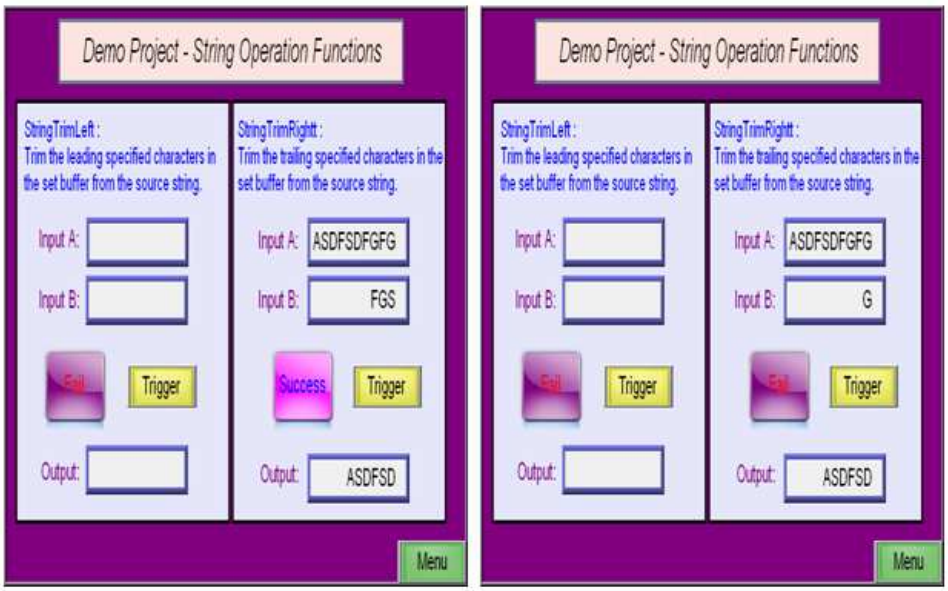
StringTrimLeft

Use StringTrimLeft to trim [Input A] string. It starts searching from the first character of [Input A] string and trims away the characters identical to those in [Input B] string. It ends when finding a character that does not exit in [Input B] string. If the result exceeds the set number of characters, in this demonstration it's 8, "Fail" indicator will be shown.



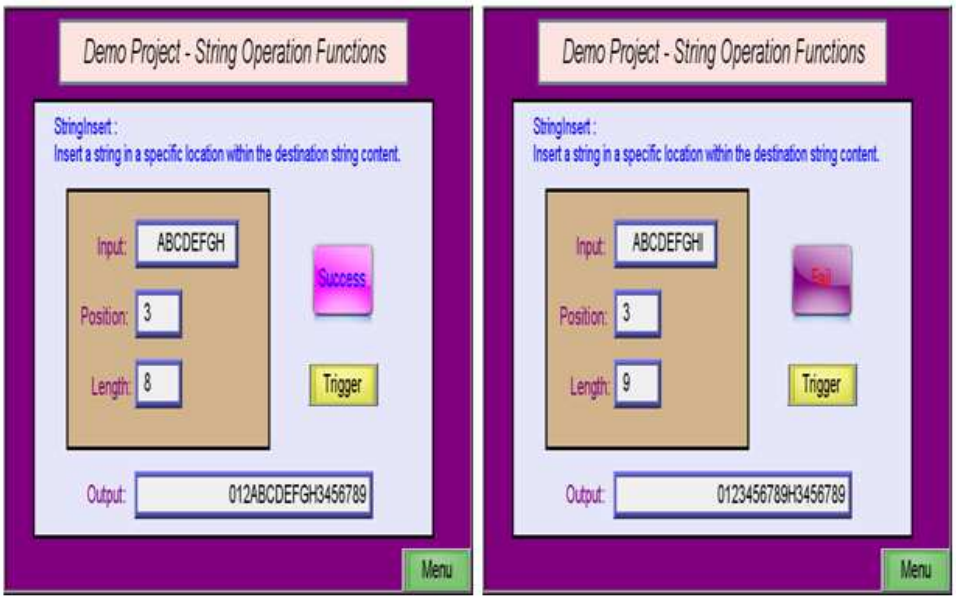
StringTrimRight

Use StringTrimLeft to trim [Input A] string. It starts searching from the last character of [Input A] string and trims away the characters identical to those in [Input B] string. It ends when finding a character that does not exit in [Input B] string. If the result exceeds the set number of characters, in this demonstration it's 8, "Fail" indicator will be shown.



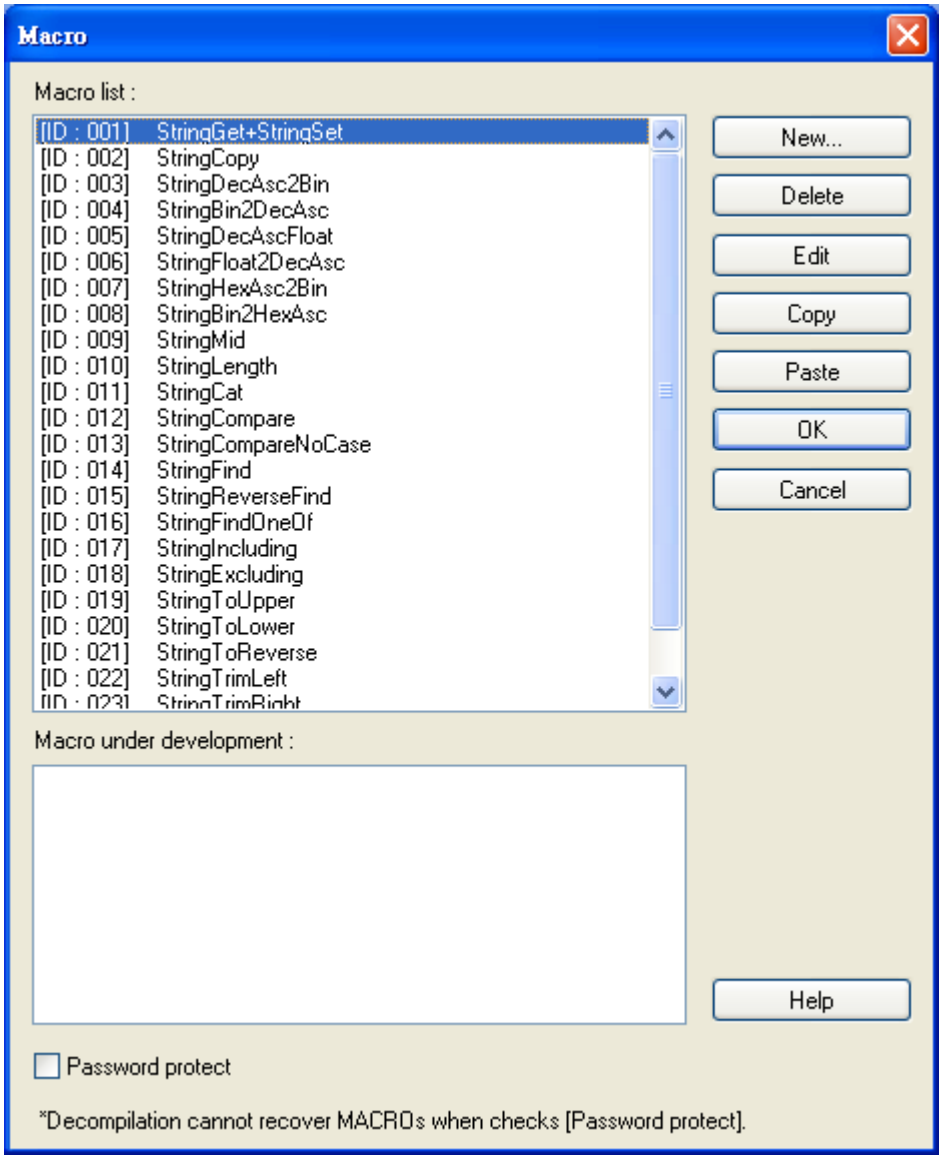
StringInsert

Use StringInsert to insert the input string into the default string “0123456789”. The position and string length to insert are defined in [Position] and [Length] fields. If the result exceeds the set number of characters, in this demonstration it’s 18, “Fail” indicator will be shown.



2 Setting Up the Screen

2-1 Edit the String Operation Functions of Macro.



[ID:001]StringGet+StringSet:

```
1
2   macro_command main()
3
4   char str1[10]
5
6   StringGet(str1[0], "Local HMI", LW, 0, 10)
7   StringSet(str1[0], "Local HMI", LW, 10, 10)
8
9   end macro_command
```

[ID:002]StringCopy:

```
1
2   macro_command main()
3
4   char src[10]
5   char dest[8]
6   bool result
7   StringGet(src[0], "Local HMI", LW, 0, 10)
8   result=StringCopy(src[0],dest[0])
9   StringSet(dest[0], "Local HMI", LW, 20, 10)
10  SetData(result, "Local HMI", LB, 0, 1)
11
12  end macro_command
```

[ID:003]StringDecAsc2Bin:

```
1
2   macro_command main()
3
4   char src[10]
5   int result
6   bool success
7
8   StringGet(src[0], "Local HMI", LW, 30, 10)
9   success = StringDecAsc2Bin(src[0], result)
10  SetData(result, "Local HMI", LW, 40, 1)
11  SetData(success, "Local HMI", LB, 1, 1)
12
13  end macro_command
```

[ID:004]StringBin2DecAsc:

```
1
2   macro_command main()
3
4   char src[10]
5   int result
6   bool success
7
8   StringGet(src[0], "Local HMI", LW, 30, 10)
9   success = StringDecAsc2Bin(src[0], result)
10  SetData(result, "Local HMI", LW, 40, 1)
11  SetData(success, "Local HMI", LB, 1, 1)
12
13  end macro_command
```

[ID:005]StringDecAscFloat:

```
1
2   macro_command main()
3
4   char src[10]
5   int result
6   bool success
7
8   StringGet(src[0], "Local HMI", LW, 30, 10)
9   success = StringDecAsc2Bin(src[0], result)
10  SetData(result, "Local HMI", LW, 40, 1)
11  SetData(success, "Local HMI", LB, 1, 1)
12
13  end macro_command
```

[ID:006]StringFloat2DecAsc:

```
1
2   macro_command main()
3
4   float src
5   char dest[8]
6   bool success
7
8   GetData(src, "Local HMI", LW, 70, 1)
9   success = StringFloat2DecAsc(src, dest[0])
10  StringSet(dest[0], "Local HMI", LW, 80, 10)
11  SetData(success, "Local HMI", LB, 4, 1)
12
13  end macro_command
```

[ID:007]StringHexAsc2Bin:

```
1
2   macro_command main()
3
4   char src[10]
5   int result
6   bool success
7
8   StringGet(src[0], "Local HMI", LW, 90, 10)
9   success = StringHexAsc2Bin(src[0], result)
10  SetData(result, "Local HMI", LW, 100, 1)
11  SetData(success, "Local HMI", LB, 5, 1)
12
13  end macro_command
```

[ID:008]StringBin2HexAsc:

```
macro_command main()

int src
char dest[5]
bool success

GetData(src, "Local HMI", LW, 100, 1)
success = StringBin2HexAsc(src, dest[0])
StringSet(dest[0], "Local HMI", LW, 110, 10)
SetData(success, "Local HMI", LB, 6, 1)

end macro_command
```

[ID:009]StringMid:

```
macro_command main()

char src[10]
char dest[8]
short a,b
bool success

StringGet(src[0], "Local HMI", LW, 120, 10)
GetData(a, "Local HMI", LW, 130, 1)
GetData(b, "Local HMI", LW, 131, 1)
success = StringMid(src[a],b, dest[0])
StringSet(dest[0], "Local HMI", LW, 140, 10)
SetData(success, "Local HMI", LB, 7, 1)

end macro_command
```

[ID:010]StringLength:

```
macro_command main()

char src[10]
short a,length


StringGet(src[0], "Local HMI", LW, 150, 10)
GetData(a, "Local HMI", LW, 156, 1)

length=StringLength(src[a])

SetData(length, "Local HMI", LW, 158, 1)

end macro_command
```

[ID:011]StringCat:

```
macro_command main()

char src1[10],src2[8]
short a
bool success

StringGet(src1[0], "Local HMI", LW, 160, 10)
StringGet(src2[0], "Local HMI", LW, 165, 10)

GetData(a, "Local HMI", LW, 170, 1)

success = StringCat(src1[a],src2[0])

StringSet(src2[0], "Local HMI", LW, 180, 10)
SetData(success, "Local HMI", LB, 8, 1)

end macro_command
```

[ID:012]StringCompare:

```
macro_command main()

char inputa[10],inputb[10]
bool ret

StringGet(inputa[0], "Local HMI", LW, 190, 10)
StringGet(inputb[0], "Local HMI", LW, 195, 10)
ret = StringCompare(inputa[0],inputb[0])

SetData(ret, "Local HMI", LB, 9, 1)

end macro_command
```

[ID:013]StringCompareNoCase:

```
macro_command main()

char inputa[10],inputb[10]
bool ret

StringGet(inputa[0], "Local HMI", LW, 190, 10)
StringGet(inputb[0], "Local HMI", LW, 195, 10)
ret = StringCompareNoCase(inputa[0],inputb[0])

SetData(ret, "Local HMI", LB, 10, 1)

end macro_command
```

[ID:014]StringFind:

```
1
2   macro_command main()
3
4   char src[10]
5   char target[10]
6   short pos
7
8   StringGet(src[0], "Local HMI", LW, 200, 10)
9   StringGet(target[0], "Local HMI", LW, 205, 10)
10  pos = StringFind(src[0],target[0])
11
12  SetData(pos, "Local HMI", LW, 210, 1)
13
14  end macro_command
```

[ID:015]StringReverseFind:

```
1
2   macro_command main()
3
4   char src[10],target[10]
5   short pos
6
7   StringGet(src[0], "Local HMI", LW, 200, 10)
8   StringGet(target[0], "Local HMI", LW, 205, 10)
9   pos = StringReverseFind(src[0],target[0])
10
11  SetData(pos, "Local HMI", LW, 211, 1)
12
13  end macro_command
```

[ID:016]StringFindOneOf:

```
1
2   macro_command main()
3
4   char src[10]
5   char target[10]
6   short pos
7
8   StringGet(src[0], "Local HMI", LW, 200, 10)
9   StringGet(target[0], "Local HMI", LW, 205, 10)
10  pos = StringFindOneOf(src[0],target[0])
11
12  SetData(pos, "Local HMI", LW, 212, 1)
13
14  end macro_command
```

[ID:017]StringIncluding:

```
macro_command main()

char inputa[10],inputb[10],output[8]
bool success

StringGet(inputa[0], "Local HMI", LW, 220, 10)
StringGet(inputb[0], "Local HMI", LW, 225, 10)
success = StringIncluding(inputa[0],inputb[0],output[0])

StringSet(output[0], "Local HMI", LW, 230, 10)
SetData(success, "Local HMI", LB, 11, 1)

end macro_command
```

[ID:018]StringExcluding:

```
macro_command main()

char inputa[10],inputb[10],output[8]
bool success

StringGet(inputa[0], "Local HMI", LW, 240, 10)
StringGet(inputb[0], "Local HMI", LW, 245, 10)
success = StringExcluding(inputa[0],inputb[0],output[0])

StringSet(output[0], "Local HMI", LW, 250, 10)
SetData(success, "Local HMI", LB, 12, 1)

end macro_command
```

[ID:019]StringToUpper:

```
macro_command main()

char input[10],output[8]
bool success

StringGet(input[0], "Local HMI", LW, 260, 10)

success = StringToUpper(input[0],output[0])

StringSet(output[0], "Local HMI", LW, 270, 10)
SetData(success, "Local HMI", LB, 13, 1)

end macro_command
```

[ID:020]StringToLower:

```
macro_command main()

char input[10],output[8]
bool success

StringGet(input[0], "Local HMI", LW, 280, 10)

success = StringToLower(input[0],output[0])

StringSet(output[0], "Local HMI", LW, 290, 10)
SetData(success, "Local HMI", LB, 14, 1)

end macro_command
```

[ID:021]StringToReverse:

```
macro_command main()

char input[10],output[8]
bool success

StringGet(input[0], "Local HMI", LW, 300, 10)

success = StringToReverse(input[0],output[0])

StringSet(output[0], "Local HMI", LW, 310, 10)
SetData(success, "Local HMI", LB, 15, 1)

end macro_command
```

[ID:022]StringTrimLeft:

```
macro_command main()

char inputa[10],inputb[10],output[8]
bool success

StringGet(inputa[0], "Local HMI", LW, 320, 10)
StringGet(inputb[0], "Local HMI", LW, 325, 10)
success = StringTrimLeft(inputa[0],inputb[0],output[0])

StringSet(output[0], "Local HMI", LW, 330, 10)
SetData(success, "Local HMI", LB, 16, 1)

end macro_command
```

[ID:023]StringTrimRight:

```
1
2   macro_command main()
3
4   char inputa[10],inputb[10],output[8]
5   bool success
6
7   StringGet(inputa[0], "Local HMI", LW, 340, 10)
8   StringGet(inputb[0], "Local HMI", LW, 345, 10)
9   success = StringTrimRight(inputa[0],inputb[0],output[0])
10
11  StringSet(output[0], "Local HMI", LW, 350, 10)
12  SetData(success, "Local HMI", LB, 17, 1)
13
14  end macro_command
```

[ID:024]StringInsert:

```
1
2   macro_command main()
3
4   char input[10]
5   short pos
6   short length
7   char output[18]="0123456789"
8   bool success
9
10  StringGet(input[0], "Local HMI", LW, 360, 10)
11  GetData(pos, "Local HMI", LW, 365, 1)
12  GetData(length, "Local HMI", LW, 366, 1)
13
14  success = StringInsert(pos, input[0], length, output[0])
15  StringSet(output[0], "Local HMI", LW, 370, 20)
16  SetData(success, "Local HMI", LB, 18, 1)
17
18  end macro_command
```

2-2 Create a Function Key to trigger Macro.

3 Addresses

The Object Addresses used in this demo project are listed below: Users can change Addresses and Object ID base on actual usage.

Addresses		Object ID	Detail
Window 4			
Function Key		FK_0	Change full-screen window to 30
Window 10			
Bit Lamp	LB-0	BL_0	Macro2 Result Indicator
ASCII Input	LW-0~9	AE_0	Macro1&2's Input
	LW-10~19	AE_1	Macro1's Output
	LW-20~29	AE_2	Macro2's Output
Function Key		FK_0	To trigger macro 1
		FK_2	To trigger macro 2
Window 11			
Bit Lamp	LB-1	BL_0	Macro3 Result Indicator
	LB-2	BL_1	Macro4 Result Indicator
ASCII Input	LW-30~39	AE_0	Macro3's Input
	LW-50~59	AE_1	Macro4's Output
Numeric Input	LW-40	NE_0	Macro3's Output
	LW-40	NE_1	Macro4's Input
Function Key		FK_0	To trigger macro 3
		FK_1	To trigger macro 4
Window 12			
Bit Lamp	LB-3	BL_0	Macro5 Result Indicator
	LB-4	BL_1	Macro6 Result Indicator
ASCII Input	LW-60~69	AE_0	Macro5's Input
	LW-80~89	AE_1	Macro6's Output
Numeric Input	LW-70	NE_0	Macro5's Output
	LW-70	NE_1	Macro6's Input
Function Key		FK_0	To trigger macro 5
		FK_1	To trigger macro 6
Window 13			
Bit Lamp	LB-5	BL_0	Macro7 Result Indicator
	LB-6	BL_1	Macro8 Result Indicator

ASCII Input	LW-110~119	AE_0	Macro8's Output
	LW-90~99	AE_1	Macro7's Input
Numeric Input	LW-100	NE_0	Macro7's Output
	LW-100	NE_1	Macro8's Input
Function Key		FK_0	To trigger macro 7
		FK_1	To trigger macro 8
Window 14			
Bit Lamp	LB-7	BL_1	Macro9 Result Indicator
ASCII Input	LW-120~129	AE_0	Macro9's Input
	LW-140~149	AE_1	Macro9's Output
Numeric Input	LW-130	NE_0	String Start Position
	LW-131	NE_1	String Length
Function Key		FK_0	To trigger macro 9
Window 15			
ASCII Input	LW-150~154	AE_0	Macro10's Input
Numeric Input	LW-156	NE_0	String Start Position
	LW-158	NE_1	Macro10's Output
Function Key		FK_0	To trigger macro 10
Window 16			
Bit Lamp	LB-8	BL_0	Macro11 Result Indicator
ASCII Input	LW-160~164	AE_0	Macro11's Input A
	LW-165~170	AE_1	Macro11's Input B
	LW180~189	AE_2	Macro11's Output
Numeric Input	LW-170	NE_0	String Start Position
Function Key		FK_0	To trigger macro 11
Window 17			
Bit Lamp	LB-9	BL_0	Macro12 Result Indicator
	LB-10	BL_1	Macro13 Result Indicator
ASCII Input	LW-190~194	AE_0	Macro12&13's Input A
	LW-195~199	AE_1	Macro12&13's Input B
Function Key		FK_0	To trigger macro 12
		FK_1	To trigger macro 13
Window 18			
ASCII Input	LW-200~204	AE_0	Macro14&15&16's Input A
	LW-205~299	AE_1	Macro14&15&16's Input B

Numeric Input	LW-210	NE_0	Macro14's Output
	LW-211	NE_1	Macro15's Output
	LW-212	NE_2	Macro16's Output
Function Key		FK_0	To trigger macro 14
		FK_1	To trigger macro 15
		FK_2	To trigger macro 16
Window 19			
Bit Lamp	LB-11	BL_1	Macro17 Result Indicator
ASCII Input	LW-220~224	AE_0	Macro17's Input A
	LW-225~229	AE_1	Macro17's Input B
	LW-230~234	AE_2	Macro17's Output
Function Key		FK_0	To trigger macro 17
Window 20			
Bit Lamp	LB-12	BL_0	Macro18 Result Indicator
ASCII Input	LW-240~244	AE_0	Macro18's Input A
	LW-245~249	AE_1	Macro18's Input B
	LW-250~254	AE_2	Macro18's Output
Function Key		FK_1	To trigger macro 18
Window 21			
Bit Lamp	LB-13	BL_0	Macro19 Result Indicator
	LB-14	BL_1	Macro20 Result Indicator
	LB-15	BL_2	Macro21 Result Indicator
ASCII Input	LW-260~264	AE_0	Macro19's Input
	LW-270~274	AE_1	Macro19's Output
	LW-280~284	AE_2	Macro20's Input
	LW-290~294	AE_3	Macro20's Output
	LW-300~304	AE_4	Macro21's Input
	LW-310~314	AE_5	Macro21's Output
Function Key		FK_0	To trigger macro 19
		FK_1	To trigger macro 20
		FK_2	To trigger macro 21
Window 22			
Bit Lamp	LB-16	BL_0	Macro22 Result Indicator
	LB-17	BL_2	Macro23 Result Indicator
ASCII Input	LW-320~324	AE_0	Macro22's Input A
	LW-325~329	AE_1	Macro22's Input B

	LW-330~334	AE_2	Macro22's Output
	LW-340~344	AE_3	Macro23's Input A
	LW-345~349	AE_4	Macro23's Input B
	LW-350~354	AE_5	Macro23's Output
Function Key		FK_0	To trigger macro 22
		FK_1	To trigger macro 23
Window 23			
Bit Lamp	LB-18	BL_0	Macro24 Result Indicator
ASCII Input	LW-360~364	AE_0	Macro24's Input
	LW-370~379	AE_1	Macro24's Output
Numeric Input	LW-365	NE_0	Position
	LW-366	NE_2	Length
Function Key		FK_0	To trigger macro 24
Window 30			
Function Key		FK_0	Change full-screen window to 10
		FK_1	Change full-screen window to 11
		FK_2	Change full-screen window to 12
		FK_3	Change full-screen window to 13
		FK_4	Change full-screen window to 14
		FK_5	Change full-screen window to 15
		FK_6	Change full-screen window to 16
		FK_7	Change full-screen window to 17
		FK_8	Change full-screen window to 18
		FK_9	Change full-screen window to 19
		FK_10	Change full-screen window to 20
		FK_11	Change full-screen window to 21
		FK_12	Change full-screen window to 22
		FK_13	Change full-screen window to 23