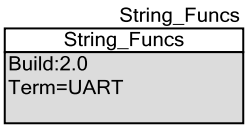
String_Funcs
2.0.3

Features



- Selected String Functions tied to a Terminal component.
- Filtering of input characters based on filter type selected.

General description

The String_Funcs Component implements a simple API interface to the designer to capture input from a Terminal device.

The current String Functions implemented:

- Two Filtered GetChar() equivalent functions (with and without control characters)
- Filtered GetString() function. The designer can select which input string type is to input from the Terminal device. See supported input_filter_types.

Although this component can be installed separately in a TopDesign schematic it must be used with a Terminal component such as the Cypress UART or USBUART. For this reason, if you choose to use the custom “Term” component it already has this component embedded in it to simplify the design of its use in an Application.

When to use String_Funcs component

The String_Funcs Component was developed to simplify the design of Terminal String handling.

Use of this custom component requires the designer to configure the project to reference the component library supplied. Refer to "How to Access Custom Libraries and Components" for methods of access available.

String_Funcs Requirements

The String_Funcs component is tied to a Terminal component in the PsoC design. To complete this link to the Terminal component, you must:

- Allocate a Terminal (ie UART or USBUART) component. In this component we have provided a “Term” component that can be configured very easily to be a UART-style or USBUART-style.

- Pointers to the _GetChar() and _PutChar() API calls of the selected Terminal component need to be installed as callbacks before use of the functions. Here is an example:
  String_Funcs _SetCallbacks((void *)&Term_GetChar, (void *)&Term_PutChar);
  If you are using the String_Funcs component directly in the TopDesign, if you supply the instanced name of the Terminal component in the "Terminal name" field of this component you can substitute the String_Funcs _Init()or String_Funcs _Start() in place of the String_Funcs _SetCallbacks() call. These API calls will load the appropriate callback pointers.
- The Terminal Program used on the Host computer should have "NO LOCAL ECHO" setting selected. The String_Funcs will echo characters back to the Terminal Device if the filtering criteria is satisfied for the character inputted.
- String_Funcs_GetString_Filt() supports the BACKSPACE character by removing the last character entered from the result string. However, the BACKSPACE character is NOT ECHOed back to the Terminal. Therefore your Terminal program may still display the character deleted in the result string.

Device Families Supported

At this time, all PsoC devices support this component. It has been only tested on PsoC5 and PsoC6 devices. This component was intended to be generic and not PsoC device specific. Theoretically, this code could be portable to ANY system supporting C code.

Implementation Limitations

This version of the String_Funcs component should satisfy most Application needs. The Filtering types supported (listed above) can be expanded or modified as needed. The C source code is included in this component. New filter types can be added and the printable characters are defined in an array where filtering criteria are bit defines.

NOTE: The code provided has no warranties expressed or implied. It is intended for non-critical or non-safety use and can be used for educational purposes.

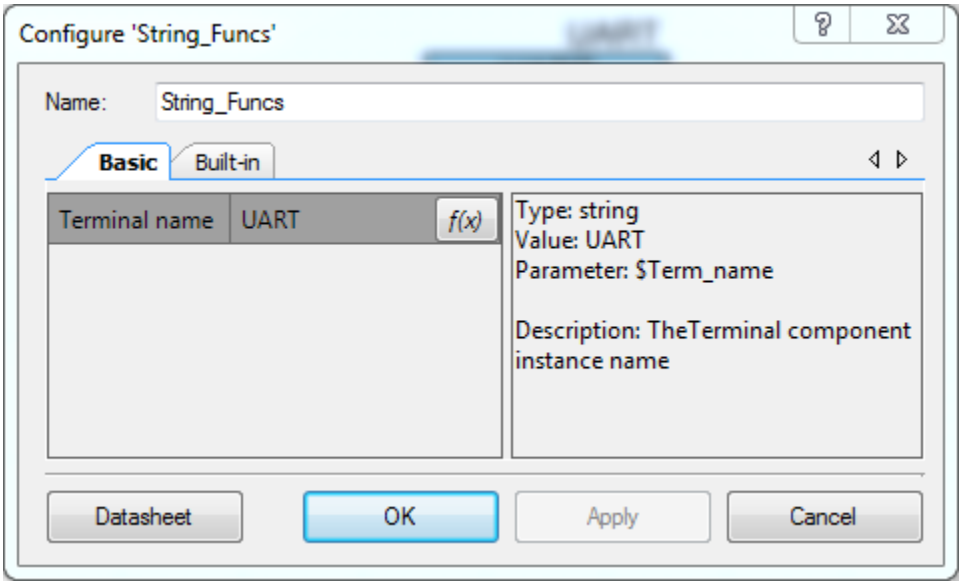
The return from String_Funcs_GetString() is a filtered string. The order of characters is as the user entered them. At this time, there is no reordering of characters such as the decimal point, plus or minus signs or the exponent character. It is up to the Application to correctly parse the result string. This can usually be done using a sscanf() function.

The _Get___Filt() API functions are implemented as BLOCKING functions. When executing these functions, the function will not return to the application until a the CR ('\r') is entered by the user. A future release may include RTOS-friendly non-blocking versions of these functions.

Parameters and Settings

Here is the parameter setting in the Configuration for String_Funcs component.

Basic tab



This tab provides following parameters:

Terminal name

- Type the name of the Terminal UART component instance name. This name is used to associate the Terminal component name for the String_Funcs_Init() or String_Funcs_Start() functions. For example: if you use the Cypress UART component and its name is "UART_1", type the name in this field.

Application Programming Interface (API)

To use this component, you must:

- Drag the String_Funcs component on the TopDesign schematic.

- Configure the component by typing in the Terminal UART component instance name in the “Terminal name” field.

These are enum types and the API calls supported:

Function	Description
input_filter_types	Input Filter Type names
String_Funcs_GetString_Filt()	Get Terminal input Strings with selectable filters
String_Funcs_GetChar_Filtcc()	Get a single character (includes BACKSPACE and CR control characters)
String_Funcs _GetChar_Filt()	Get a single printable character (no control chars)
String_Funcs _SetCallbacks()	Set CallBacks for Terminal GetChar() and PutChar()
String_Funcs _Init()	Call to _ SetCallbacks.
String_Funcs _Start()	Call to _ SetCallbacks.

input_filter_types

Input Filter Type	Description	Example
ift_dec	Unsigned Decimal (positive-only base 10 integers)	1045
ift_sdec	Signed Decimal (positive or negative base 10 integers)	-943
ift_dec_fixed	Unsigned Fixed Decimal (positive-only base 10 fractional numbers)	3.1415
ift_sdec_fixed	Signed Fixed Decimal (positive or negative base 10 fractional numbers)	-9.43
ift_hex	Hexidecimal (base 16 integers)	0A2F45
ift_string	String (printable string characters)	String01
ift_str_fs	FileSystem String (selected string characters allowed in filesystem naming)	A:FS\01
ift_float	Float (A positive or negative floating number. Ie. 15.67 or -93.34e10)	3.1415e1

uint8_t GetString_Filt(char string[], uint16_t string_sz, input_filter_types filter)

Function to gets the Terminal device input with the selected filtering. Note: The CR (‘\r’) is used to terminate the Terminal input capture. The NULL (‘\0’) is appended to the end of the supplied input string to provide a valid string for further processing.

The supplied input string size MUST NOT exceed the allocated input string. This parameter is intended to prevent overwriting the string creating a ‘safe’ string handling function.

NOTE: This is a blocking function.

Parameters:

string	Pointer to input string. Must be large enough to hold the desired string

string_sz	input string size. Must not exceed the input string allocated.
filter	Input Filter Type.

Returns:
The number of characters in the supplied input string. (Does not include the CR character or the NULL '\0' character terminating the string.)

Example:

Char string[20];

Get_String_Filt(string, sizeof(string), ift_dec);

uint8_t GetChar_Filtcc(input_filter_types filter, _Bool allow_cc)

Function to gets a char from the Terminal device input with the selected filtering. If the filter criteria for the input character is not met, the function does not return.

BACKSPACE ('\b') and CR ('\r') Control characters are NOT filtered out.

NOTE: This is a blocking function.

Parameters:

filter	Input Filter Type.

Returns:
The filtered character.

Example:

uint8_t c;

c = Get_Char_Filtcc(ift_hex, 1);

uint8_t GetChar_Filt(input_filter_types filter)

Function to gets a single printable char from the Terminal device input with the selected filtering. If the filter criteria for the input character is not met, the function does not return.

Control characters are filtered out.

NOTE: This is a blocking function.

NOTE: This a actually a indirect call to Get_Char_Filtcc(filter, 0);

Parameters:

filter	Input Filter Type.

Returns:
The filtered character.

Example:

```
uint8_t  c;

c = Get_Char_Filt( ift_hex);
```

int String_Funcs_SetCallbacks(uint8_t (*GetChar_cb)(void), void (*PutChar_cb)(char pc_c));

Mandatory: Initialize the String_Funcs component by installing callbacks to Terminal functions. Not issuing this function with correct pointers to _GetChar() and _PutChar() functions before use will yield no communication results.

This function allows the designer to switch to different Terminal ports at run-time if desired.

Parameters:

&_GetChar	Pointer to the Terminal's _GetChar function.
&_PutChar	Pointer to the Terminal's _PutChar function.

Returns:
error // 0 = No Error

Example:

```
        // Term_ is the TopDesign terminal component name.
String_Funcs_SetCallbacks ((void *)&Term_GetCha r, (void *)&Term_PutChar);
```

int String_Funcs_Init(void)

Use this function to initialize the String_Funcs component. It initializes the callback for _PutChar() to another Terminal instance name listed in the "Terminal name" parameter.

int String_Funcs_Start(void)

Use this function calls String_Funcs_Init().

Resources

The String_Funcs component utilizes some FLASH and SRAM resource resources. If you do not use all String_Funcs functions and you optimize your builds, unused functions will be automatically not included in the object file.

Sample Firmware Source Code

A String_Funcs_Demo project is included to demonstrate each of the supported functions of this component. The default device for this project is a PsoC5 and configured for the CY8CKIT-059 board. The project can be modified to work with other PsoC devices. It also uses the embedded instance of this component in the Term component.

The MenuCmds component is also used in this demo project to provide a easy to use single character menu command structure to execute different features.

Component Changes

Version	Description of changes	Reason for changes/impact
2.0.3	Component released version	Placed code in a shareable component.
2.0	first release of the component	Non-component version.

References

String_Funcs_Demo_v2_1_0.pdf	String_Funcs_Non_Term_Demo_v2_1_0.pdf
Term Component: Term_v2_2.pdf	MenuCmds Component: MenuCmds_v3_3.pdf
How to Access Custom Libraries and Components.pdf	