

## Term 2.3

### Features

- Simplifies the interface to a Terminal UART for a Cypress serial UART or Cypress USBUART component.
- Uses the Cypress serial UART API calls.
- Full-Duplex serial communication.
- Optional Terminal-connected String Functions like GetString\_Filt() and GetChar\_Filt().
- Optional Terminal-connected single character Menu Command structure.

### General description

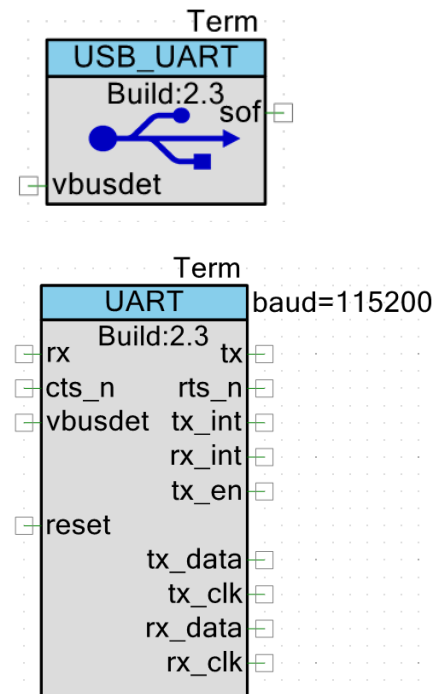
The Term Component implements a simple interface on the PsoC to a PC-based Terminal Application. The component has both the UART and USBUART components embedded in the implementation but you select which UART-type is used.

This version of Term allows the user to access all of the inputs, outputs and functional API interfaces available to the Cypress UART V2.50. Therefore, user setting of the baudrate, flow control and other features are now available.

Embedded are additional optional library/components that use the Term Component to implement:

- [String\\_Funcs](#) => GetString\_Filt() – Filters input characters that match the filter criteria passed to the function to return a string. GetChar\_Filt() - Filters input characters that match the filter criteria passed to the function to return a single character.
- [MenuCmds](#) => Allows for a structure of Menu commands using optional submenus to invoke function calls on single character input from the Terminal.

### When to use Term component



The Term Component was developed to be a “quick” change interface to use the PsoC’s UART component or the USBUART component with very minimal SW changes in an application.

For example, if you have previously designed terminal communication SW using the UART component (a very common practice), you can remove the UART component on the schematic and replace it with the Term component and give the same component name. Since it supports many of the UART component API calls, you should be able to build your project and it should work without further modification. (Make sure the Rx and Tx pin assignments are correct for your application.)

Later, if you chose to use the PsoC’s USBUART, just change the “UART type” parameter to “USB UART”. Build your project. Many of the common UART API calls are supported so that the USB port should now be useable as a UART type device.

Using the optional embedded [String\\_Funcs](#) and [MenuCmds component](#)s simplifies creating user interfaces using a Terminal host program on a PC.

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.

## Device Families Supported

At this time, the following are the PsoC devices that support this component:

- PsoC5

## Implementation Limitations

This version of the Term Component has the following limitations:

- The UART functional implementation should be complete. No known limitations.
- The USB\_UART implementation uses some of the common UART API calls but not all. If some of the USBUART component calls (or #defines) are needed, they can be accessed by renaming the call to *Term\_instance\_USBUART\_usbuart\_function*. [Table 1](#) has the UART API calls supported.

- This implementation favors using blocking API calls.
- If the “UART type” is selected as “USB UART”, then the following line of code needs to be placed in the “cyapicallbacks.h” file in the #includes section:  
`#include <Term_instance_cyapicallbacks.h>`

## UART Type Setup

Term Component was originally designed to allow applications to code for the Cypress UART V2.50. Therefore the coding and now the IO interfaces should perfectly match the Cypress UART implementation if you select the UART Type to be “UART”.

### Input-output connections

Term Component has only two IO connections (Rx and Tx) needed for serial data. **As UART type, you need to make sure the Rx and Tx pins are assigned to the correct Port and pins in the “Design Wide Resources/Pins” tab.** For example if you are using the CY8CKIT-059, the KitProg uses P12.6 for Rx and P12.7 for Tx.

In this version of the component if you use the UART type, you can request that the Rx and Tx pins be exposed for external connections. Additionally the other UART component pins can be available such as the Tx Output Enable, and the CRC data support pins.

### SW Considerations

There are no known special SW considerations for the “UART” type.

### SW Considerations

However, the inherent USBUART calls are not 100% compatible to the UART API calls and the USBUART allows for more low-level functionality. The version of the Term Component has a limited set of Cypress UART V2.50 API-equivalent calls. These are the most used calls of the UART.

### Parameters and Settings

Most of the parameters and settings of this component are identical to the Cypress UART v2.50. Refer to [UART component](#) for further information of internal implementation.

## Basic tab

Configure 'Term'

Name: Term

Basic | Advanced | UART Params | UART Params (Adv) | Built-in

Configuration Type		
UART Type	UART	f(x)
UART IO config		
Expose rx & tx pins	<input type="checkbox"/>	f(x)
Flow Control (rts_n & cts_n)	Hardware	f(x)
HW TX output enable (tx_en)	<input type="checkbox"/>	f(x)
CRC outputs enable	<input type="checkbox"/>	f(x)
Embedded Components		
Enable MenuCmds	<input checked="" type="checkbox"/>	f(x)
Enable String_Funcs	<input checked="" type="checkbox"/>	f(x)

Type: UARTType  
Value: 1  
Value name: UART  
Value display name: UART  
Parameter: \$Term\_type  
Description: Select the UART type

Datasheet OK Apply Cancel

This tab provides following parameters:

### UART type

Sets UART type to use.

- UART. This is the Cypress UART v2.50 implementation.
- USB UART. This selects the USBUART v3.20 component.

**Expose rx & tx pins** (This parameter is not available for UART type = USB UART.)

- Normally the RX and TX pins are allocated in the Term component and are normally hidden by default. However, by selecting the option, the user can expose these pins external to the component.

**Flow Control (rts\_n & cts\_n)** (This parameter is not available for UART type = USB UART.)

- None. No flow control and the cts\_n and rts\_n pins are not exposed
- Hardware. Flow control is used with the and cts\_n and rts\_n pins.

**HW TX output enable (tx\_en)** (This parameter is not available for UART type = USB UART.)

- Selecting this option allows the tx\_en to be exposed and the pin is active high when Term component is transmitting.

**CRC outputs enable (tx\_data, tx\_clk, rx\_data, rx\_clk)** (This parameter is not available for UART type = USB UART.)

- Selecting this option allows the data and clocks for both Tx and Rx for use with the serial CRC calculation component.

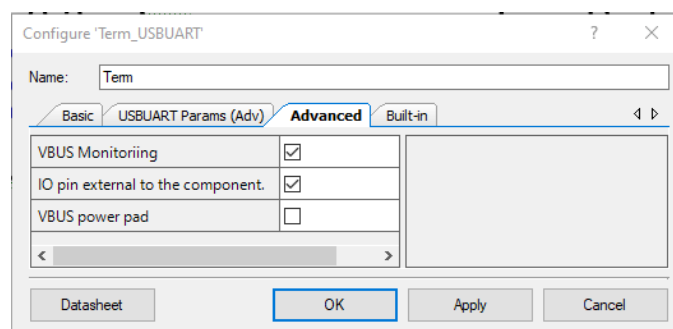
### Enable MenuCmds

- Selecting this option (default) enables the embedded component code for MenuCmds to be available to the Term Component. If you were going to use multiple Terminal ports for the user interface, disable this option and you can access the [MenuCmds component](#) by dropping it directly into your TopDesign schematic page. Then you can use the [MenuCmds component](#) to switch between different Terminal ports.

### Enable String\_Funcs

- Selecting this option (default) enables the embedded component code for String\_Funcs to be available to the Term Component. If you were going to use multiple Terminal ports for the user interface, disable this option and you can access the [String\\_Funcs component](#) by dropping it directly into your TopDesign schematic page. Then you can use the [String\\_Funcs component](#) to switch between different Terminal ports.
- Note: At this time, only one instance of this component is usable. However, any component with a `_GetChar()` function can use the string functions.

## Advanced tab



### VBUS Monitoring (default = false)

- Monitoring of the VBUS voltage detection for the USBUART type. This allows for detection of the USB cable to be connected or disconnected. This assumes that the PSoC is not being powered by this USB connection.

- This monitoring is available for UART type. Note: The standard UART communication inherently does use a powered cable like the USB\_UART. This is available for a creative user who may want to construct an equivalent. For example, some older UART standards have used signal such as DTR or DCE (look it up) to infer that the far-end of the communication is powered or available for communication. Other ideas would be creating a UDB-based HW state machine that would time-out if an expected “heart-beat” from the other end was no longer available. Be creative if you want to use it.

**IO pin external to the component (default = false = internal)**

- This parameter enables external **vbusdet** input. If not selected, the **vbusdet** input is internal to the component. This selection is only available if **VBUS Monitoring** = true.
- **Note:** Whether you chose internal or external you must select the VBUS detection pin from the “Pins” tab in the DWR. Because a cable connection or disconnection can occur while the PSoC is powered, this is considered “Hot-swappable”. It is a good design practice to use the PSoC’s SIO pins (instead of GPIO pins) for connections to the “hot-swap” connection. This minimizes the chance of damage to the PSoC.
- Choosing an external connection allows you to place HW logic to it as well as a ISR attached to this signal in your TopDesign.

**VBUS power pad (default = false)**

- Only the PSoC 4200L has the ability use this feature with the USBUART type. Selecting this without being a PSoC4200L will yield a build error.

## UART Params tab

This tab is not available for UART type = USB UART.

Configure 'Term'

Name:

Basic Advanced **UART Params** UART Params (Adv) Built-in

Bit Definitions		
Baud Rate	115200	$f(x)$
Data Bits	8	$f(x)$
Parity	None	$f(x)$
Parity from SW enable	<input type="checkbox"/>	$f(x)$
Rx Use 2 out of 3 polling	<input checked="" type="checkbox"/>	$f(x)$
Stop Bits	1	$f(x)$
Clock Selection		
Internal Clock Enable	<input checked="" type="checkbox"/>	$f(x)$
Oversampling rate	8	$f(x)$
Tx DataPath enable	<input checked="" type="checkbox"/>	$f(x)$
Mode		
Half Duplex	<input type="checkbox"/>	$f(x)$
Rx Enable	<input checked="" type="checkbox"/>	$f(x)$
Tx Enable	<input checked="" type="checkbox"/>	$f(x)$

Datasheet OK Apply Cancel

These are commonly used parameters. Refer to [UART component](#) for further information of these parameters and settings.

## UART Params (Adv) tab

This tab is not available for UART type = USB UART.

Configure 'Term'

Name:

Basic Advanced UART Params **UART Params (Adv)** Built-in

- ☐ Break params
 

Rx Break Bit count	13
Rx Break Detect enable	<input type="checkbox"/>
Tx Break Bit count	13
- ☐ Buffer Sizes
 

Rx Buffer Size	4
Tx Buffer Size	4
- ☐ RX Address Configuration
 

Rx HW address detection mode	None
------------------------------	------
- ☐ RX Interrupts
 

Internal RX Interrupt Enable	<input type="checkbox"/>
Rx Break detected	<input type="checkbox"/>
Rx Byte Received	<input checked="" type="checkbox"/>
Rx overrun error	<input type="checkbox"/>
Rx parity error	<input type="checkbox"/>
Rx stop error	<input type="checkbox"/>
- ☐ TX Interrupts
 

Internal TX Interrupt Enable	<input type="checkbox"/>
Tx complete	<input type="checkbox"/>
Tx FIFO empty	<input type="checkbox"/>
Tx FIFO full	<input type="checkbox"/>
Tx FIFO not full	<input type="checkbox"/>

< >

Datasheet OK Apply Cancel

These are advanced parameters. These may be used to enhance the performance of the UART type configuration.

Refer to [UART component](#) for further information of these parameter and settings.



## USBUART Type Setup

Term Component's intent was to allow for easy conversion from the UART SW API calls to equivalent USBUART calls. Therefore the SW and HW interface compatibility to the USBUART is limited. It was intended that the compatibility level can cover the greater majority of application uses.

### Input-output connections

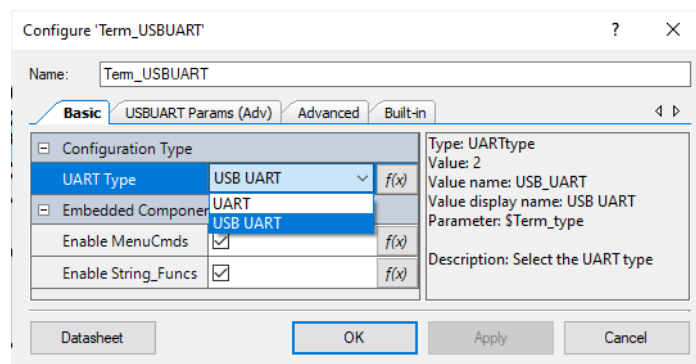
This version of the Term Component has only two IO connections (Rx and Tx) needed for serial data. As USBUART type, the Dp and Dn pins are automatically assigned to the only pins supported on the PsoC5 for this internal resource.

Future versions of the Term Component may have support for other IO connections as the need occurs.

### Parameters and Settings

There is ONLY one parameter involved in setting Term Component for UART Type = "USB UART"

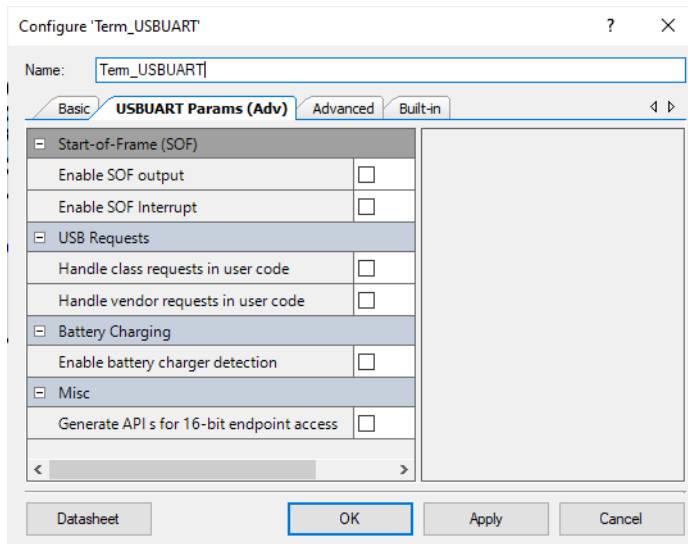
### Basic tab



Refer to UART [Basic tab](#) for other parameter definitions.

### USBUART Params (Adv) tab

This tab allows for access to USB\_UART features for HW and SW control.



### Enable SOF output (default = false)

- Selecting this will make the **sof** output available for HW synchronization to other circuits.

### Enable SOF interrupt (default = false)

- Selecting this will make the SW **sof** interrupt available to the application.

### Handle class requests in user code (default = false)

- Selecting this allows the class requests to be controlled by the application.

### Handle vendor requests in user code (default = false)

- Selecting this allows the vendor requests to be controlled by the application.

### Enable battery charger detection (default = false)

- Only the PSoC 4200L has the ability use this feature with the USBUART type. Selecting this without being a PSoC4200L will yield a build error.

### Generate APIs for 16-bits endpoint access (default = false)

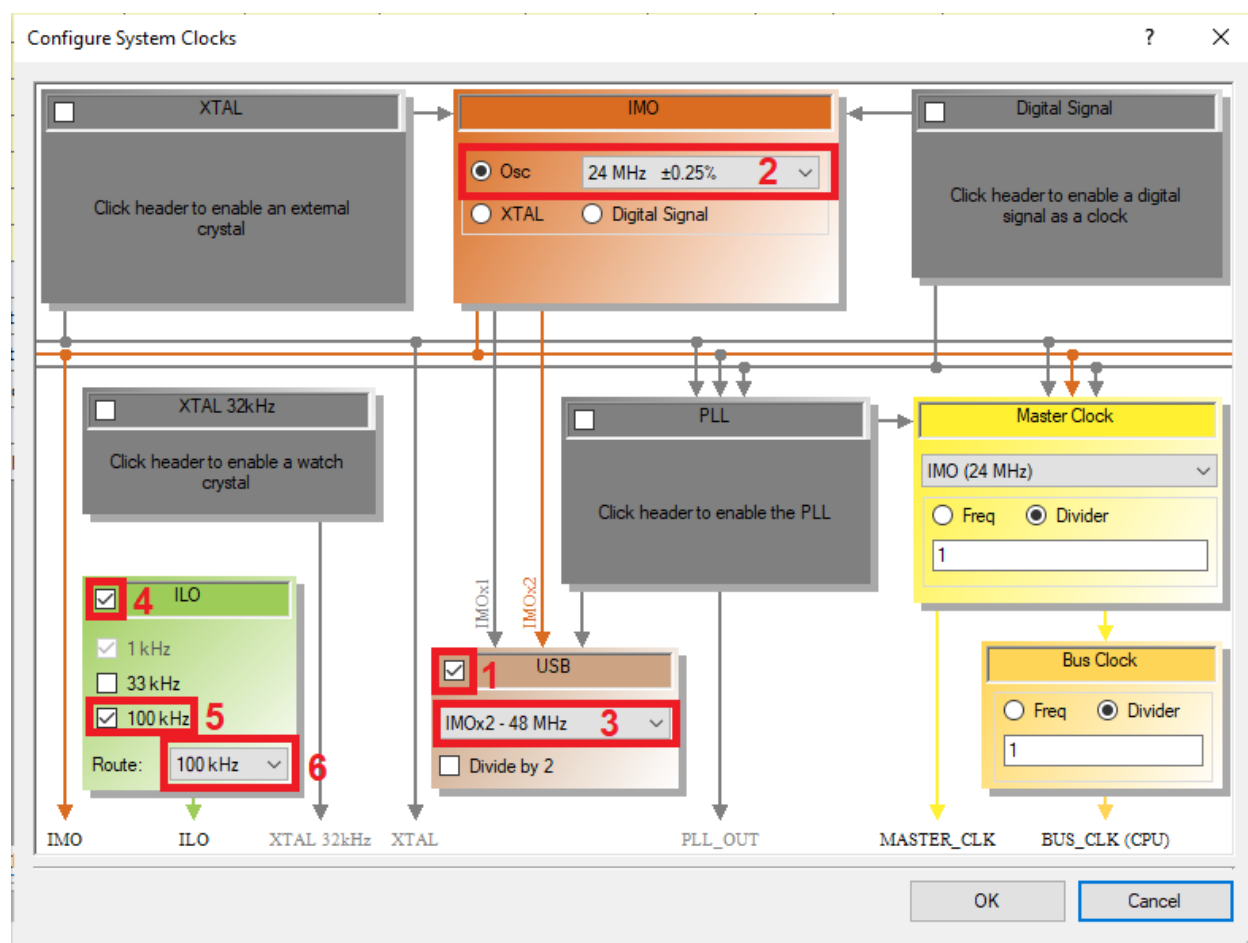
- Only the PSoC 4200L has the ability use this feature with the USBUART type. Selecting this without being a PSoC4200L will yield a build error.

## USBUART Special Application Considerations

To use the USBUART type there are a few considerations in the application that need to be implemented.

### Clock Requirements

The USBUART type needs to have some special system clocking considerations. Here is a display of the “Design Wide Resources\Clocks\Edit Clocks...” and the highlighted required clocking parameters.



1. The USB clock block enabled.
2. The internal IMO clock set to 24 MHz.
3. The USB clock set to IMOx2 – 48 MHz.
4. The ILO clock block enabled.
5. The ILO clock set to 100 KHz.
6. The ILO Route to 100 KHz.

All other clock configurations including the PLL, Master Clock and Bus Clock can be set as needed for the application.

This special USB clocking consideration can remain in place if the Term Component is switched to the UART type. The additional benefit is that if the IMO clock is used for the Bus Clock, the clock accuracy is +/- 0.25%.

### SW Requirements

The USBUART type uses a sub-set of UART API calls. Refer to [Application Programming Interface \(API\)](#) for the list of supported API calls for the USBUART type.

One more application modification needs to be performed to support the USBUART type.

The following line of code needs to be placed in the “cyapicalbacks.h” file in the #includes section:

```
#include <Term_instance_cyapicalbacks.h>
```

This additional line can remain in the “cyapicalbacks.h” file even if the Term Component is switched to the UART type.

## Application Programming Interface (API)

The API calls used for this component are identical to those used by the UART component. Refer to the UART component datasheet [UART component](#) for more information about these API calls for their arguments and return values.

Note: Not all USB UART component API calls or #defines are supported directly. If needed, the underlying component (UART or USBUART) calls or defines can be accessed by referring to it by the Term\_UART\_xxx or Term\_USBUART\_xxx (where \_xxx is the defined label).

The following table is a list of the supported API calls for each UART type.

**Table 1 - Supported API calls**

Type = UART	Type = USBUART
Term_Start	Term_Start
Term_Stop	Term_Stop
Term_ReadControlRegister	
Term_WriteControlRegister	
Term_Init	
Term_Enable	
Term_SaveConfig	
Term_RestoreConfig	
Term_Sleep	
Term_Wakeup	
Term_EnableRxInt	
Term_DisableRxInt	
Term_RXISR	
Term_SetRxAddressMode	
Term_SetRxAddress1	
Term_SetRxAddress2	
Term_SetRxInterruptMode	
Term_ReadRxData	Term_ReadRxData
Term_ReadRxStatus	
Term_GetChar	Term_GetChar
Term_GetByte	
Term_GetRxBufferSize	Term_GetRxBufferSize
Term_ClearRxBuffer	
Term_GetRxInterruptSource	
Term_EnableTxInt	
Term_DisableTxInt	

Term_SetPendingTxInt	
Term_ClearPendingTxInt	
Term_TXISR	
Term_SetTxInterruptMode	
Term_WriteTxData	
Term_ReadTxStatus	
Term_PutChar	Term_PutChar
Term_PutString	Term_PutString
Term_PutArray	Term_PutArray
Term_PutCRLF	Term_PutCRLF
Term_ClearTxBuffer	Term_ClearTxBuffer
Term_SetTxAddressMode	
Term_SendBreak	
Term_GetTxBufferSize	
Term_PutStringConst	
Term_PutArrayConst	
Term_GetTxInterruptSource	
Term_LoadRxConfig	
Term_LoadTxConfig	
Term_VBusPresent*	Term_VBusPresent

\*This function is an equivalent USBUART function.

If the embedded [String\\_Funcs component](#) is used, then here is the list of API calls:

- Term\_String\_Funcs\_SetCallbacks( )
- Term\_String\_Funcs\_Init()
- Term\_String\_Funcs\_Start()
- Term\_String\_Funcs\_GetString\_Filt()
- Term\_String\_Funcs\_GetChar\_Filt()
- Term\_String\_Funcs\_GetChar\_Filtcc()

Refer to the [MenuCmds component](#) for function input parameters and returns.

If the embedded [MenuCmds component](#) is used, then here is the list of API calls:

- Term\_MenuCmds\_SetCallbacks()
- Term\_MenuCmds\_Init()
- Term\_MenuCmds\_Start()
- Term\_MenuCmds\_process()
- Term\_MenuCmds\_num\_func\_SetCallback()

- Term\_MenuCmds\_search()

Refer to the [MenuCmds component](#) for function input parameters and returns.

## Functional Description

Term Component is a full featured UART communication component when used as UART type = UART. The USB UART is a simplified (and minimal) UART interface. If additional USBUART functionality is needed, you can directly access the underlying USBUART component.

## Resources

The Term Component utilizes the following resources.

<b>UART Type</b>	<b>Resource Type</b>							
	<b>Fixed Func</b>	<b>Datapath Cells</b>	<b>Macro cells</b>	<b>Status Cells</b>	<b>Control Cells</b>	<b>DMA Channels</b>	<b>Interrupts</b>	<b>GPIO</b>
<i>UART</i>	–	3	24	3	2	–	0	2
<i>UART w/vbusdet</i>	–	3	24	4	2	–	0	3
<i>USBUART</i>	<i>USB</i>	0	0	0	0	–	9	2
<i>USBUART w/vbusdet</i>	<i>USB</i>	0	0	0	0	–	9	3

## Sample Firmware Source Code

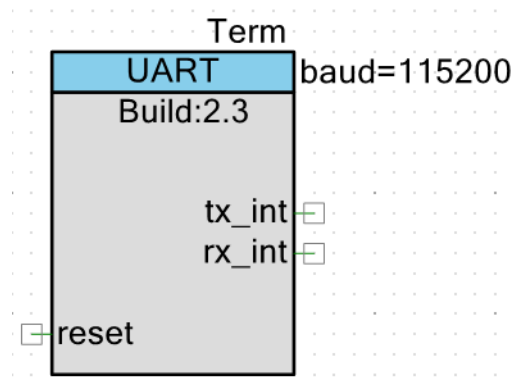
Here is some very simple sample code to write and read from the Term component.

### Expected Results

It will print off the project name and the SW version with a count that increments every second to the Terminal window. If you type characters into the Terminal window, it will be echoed to the window.

### Instructions

Drop Term component on your TopDesign schematic.



Rename the component to “Term”.

Configure for “UART Simple” or “USB UART”

←-----→

< This code goes in main.c >

```
#include "project.h"
#include "version.h"
#include <stdio.h>
#define STRINGIFY(x) #x
#define TOSTRING(x) STRINGIFY(x)

#define SW_VER_REV_REL 2.0.0

_Bool sec_1 = 0;
//-----
CY_ISR(isr_systick_0)
{
    static int16_t i = 0;
    i++;
    if(i >= 1000) {sec_1 = 1; i=0; }
}
//-----

int main(void)
{
    char tstr[100];

    CyGlobalIntEnable; /* Enable global interrupts. */
    Term_Start();

    CySysTickStart();
    CySysTickSetCallback(0,&isr_systick_0);

    for(;;)
    {
        static uint32_t count;
        char c;
        while(sec_1 == 0)
        {
            // loop while flag not set
            if( (c= Term_GetChar()) != '\0')

```



```

        {
            Term_PutChar(c);
        }
    }
    sec_1 = 0;  // reset the flag

    snprintf(tstr, sizeof(tstr), "%s %s count=%u", CY_PROJECT_NAME,
    TOSTRING(SW_VER_REV_REL), count);
    Term_PutString(tstr);
    Term_PutString("\t\t");
    count++;
}
}
/* [] END OF FILE */

```

←-----→

< This code goes into cyapicalcallbacks.h >

```

#ifdef(CYAPICALCALLBACKS_H)
#define CYAPICALCALLBACKS_H
#include <Term_cyapicalcallbacks.h>           // include other component
callbacks.

#endif /* CYAPICALCALLBACKS_H */

```

## Component Changes

Version	Description of changes	Reason for changes/impact
2.3	Minor fixes to allow for multiple instances. Added more USBUART features such as VBUS monitoring and SOF output and interrupt.	Fixed report of issue with creating multiple instances. Added request for pass-through of some USBUART features.
2.2	Full functional Cypress UART (v2.50) when UART type = UART.	Allows for more seamless use in older applications. Embedded String_Funcs and MenuCmds components
2.1	Internal release	
2.0	first release of the component	

## References

Term_Demo_v2_0_0.pdf	
UART Component: <a href="#">UART component</a>	USBUART Component: <a href="#">USBFS component</a>
String_Funcs Component: String_Funcs_v2_0.pdf	MenuCmds Component: MenuCmds_v3_3.pdf
How to Access Custom Libraries and Components.pdf	

© CONSULTRON, 2020 All Rights Reserved. CONSULTRON allows PUBLIC use of the file.

CONSULTRON allows public or commercial use of this product.

DISCLAIMER: CONSULTRON provides NO WARRANTY expressed or implied.

This product is intended for non-critical or non-safety use and can be used for educational purposes.