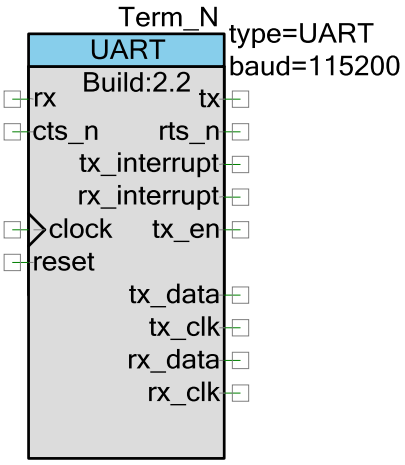
Term
2.2



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.
- Terminal-connected String Functions like GetString_Filt() and GetChar_Filt().
- 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 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 changes.

For example, if you have previously designed terminal communcation 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 usuable as a UART type device.

Using the embedded String_Funcs and MenuCmds components 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 all the USBUART component calls (or #defines) are needed, they can be accessed by renaming the call to Term_USBUART_xxx (with _xxx set to the USBUART defined label). 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_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.

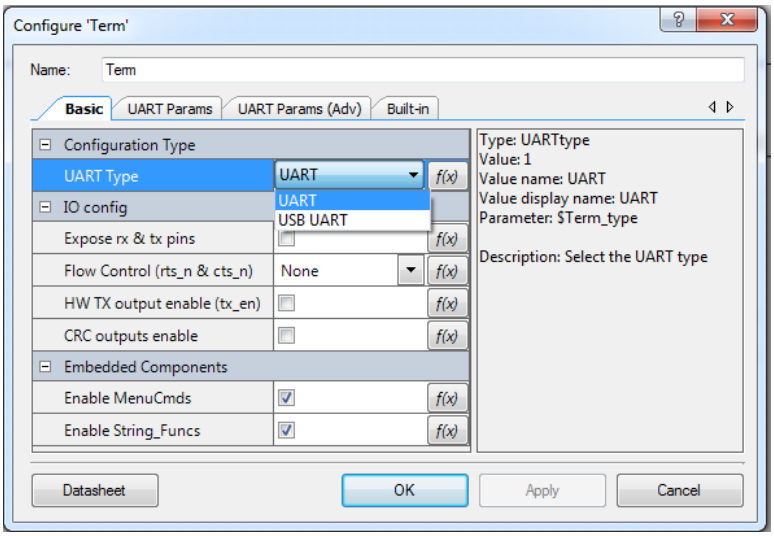
SW Considerations

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



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.

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

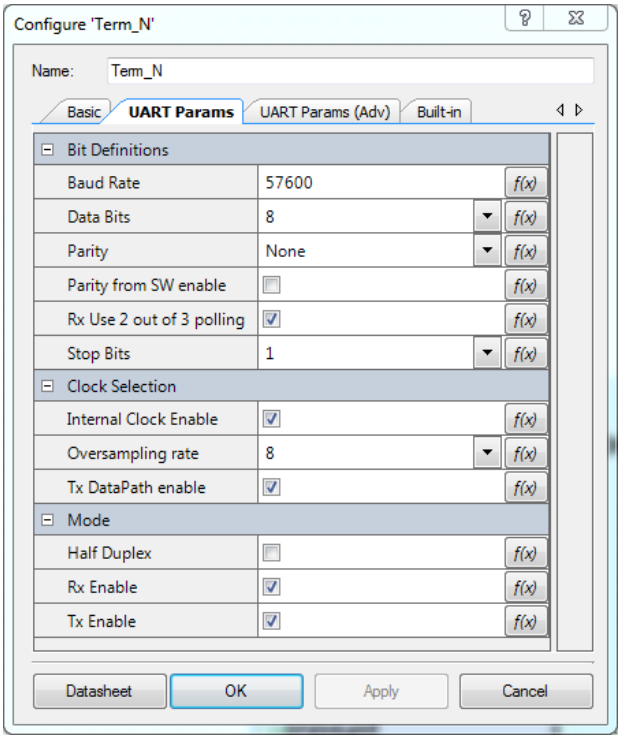
component by dropping it directly into your TopDesign schematic page. Then you can use the MenuCmds component to switch between different Terminal ports.

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.

UART Params tab

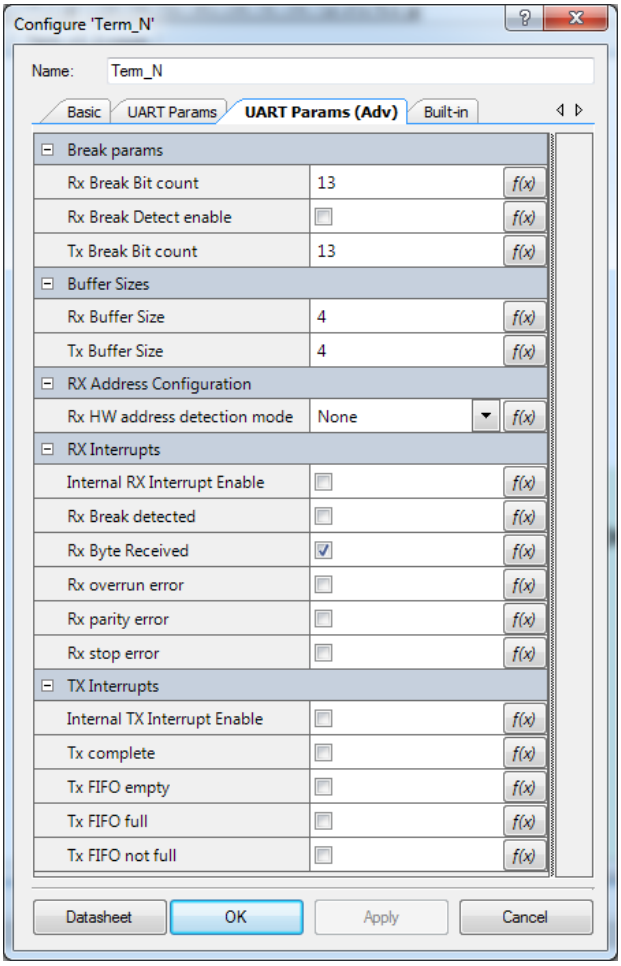
This tab is not available for UART type = USB UART.



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.



These are advanced parameters. These may be used to enhance the perfromance 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.

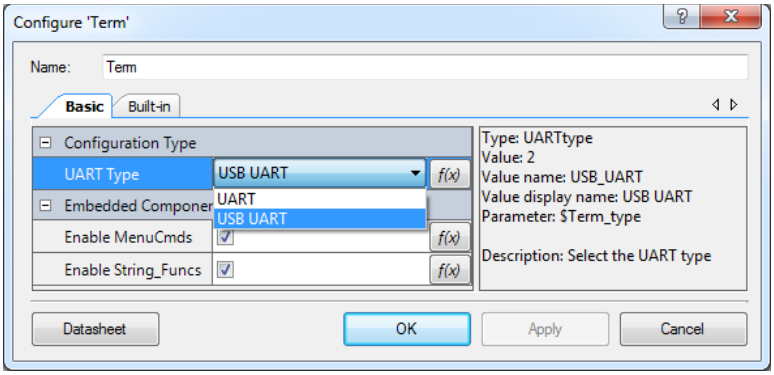
Input-output connections

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



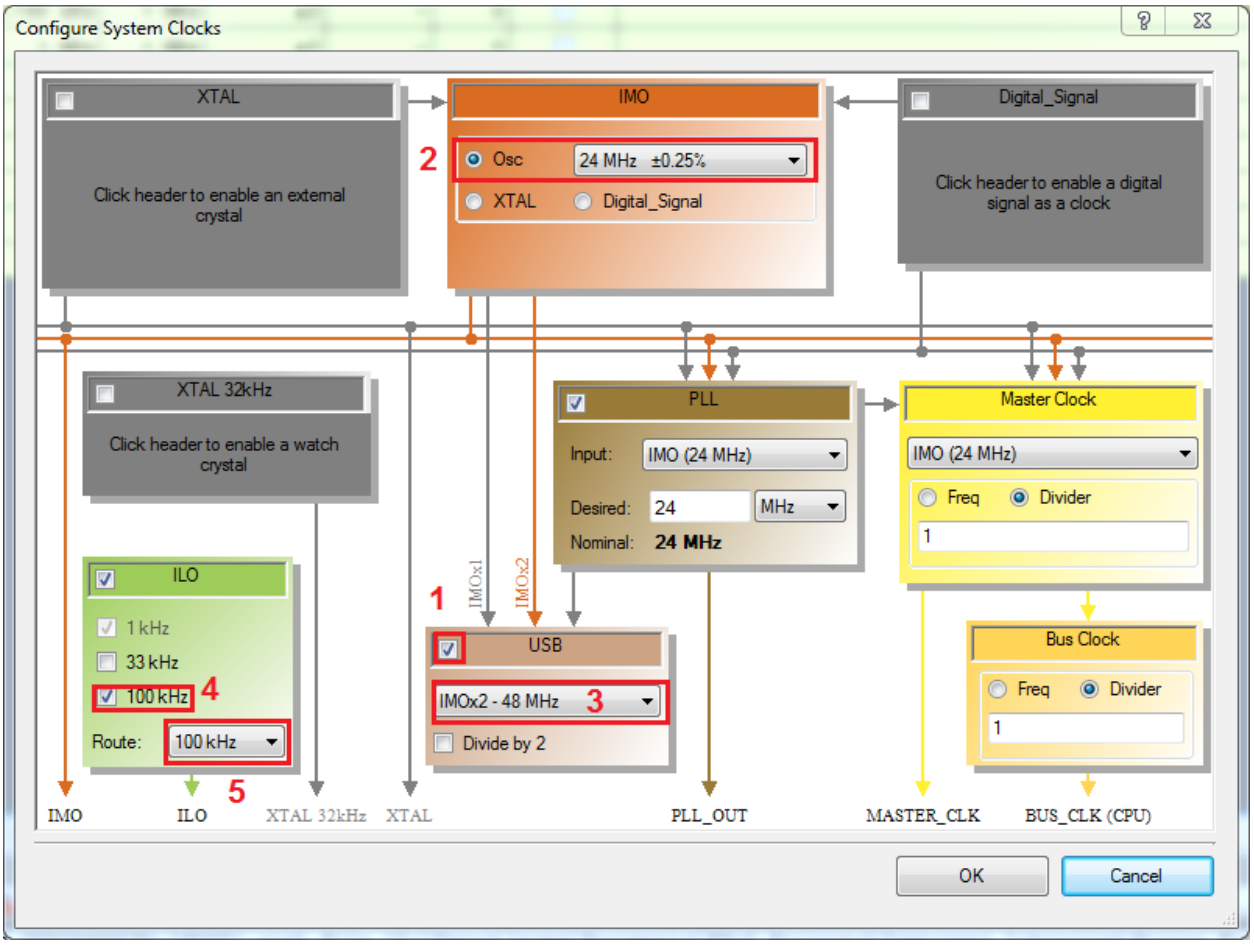
Refer to UART **Basic tab** for parameter definitions.

Special Application Considerations

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



- The USB clock block enabled.
- The internal IMO clock set to 24 MHz.
- The USB clock set to IMOx2 – 48 MHz.
- The ILO clock block enabled.
- The ILO clock set to 100 KHz.
- 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 “cyapicallbacks.h” file in the #includes section:
#include <Term_cyapicallbacks.h>

This additional line can remain in the “cyapicallbacks.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	

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 simplied (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.

	Resource Type							
UART Type	*Fixed Func*	*Datapath Cells*	*Macro cells*	*Status Cells*	*Control Cells*	*DMA Channels*	*Interrupts*	*GPIO*
UART	*–*	*3*	*24*	*3*	*1*	*–*	*0*	*2*
USBUART	*USB*	*0*	*0*	*0*	*0*	*–*	*2*	*0*

Sample Firmware Source Code

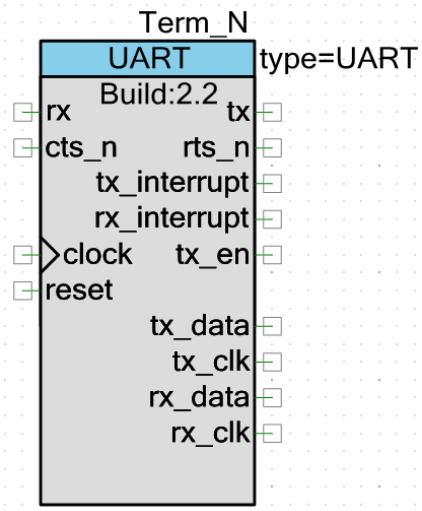
Here is some very simple sample code to write and read from the Term component.

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 cyapicallbacks.h >

```
#if !defined(CYAPICALLBACKS_H)
#define CYAPICALLBACKS_H
#include <Term_cyapicallbacks.h>          // include other component
callbacks.

#endif /* CYAPICALLBACKS_H */
```

Component Changes

Version	Description of changes	Reason for changes/impact
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: UART component	USBUART Component: USBFS component
String_Funcs Component: String_Funcs_v2_0.pdf	MenuCmds Component: MenuCmds_v3_3.pdf

How to Access Custom Libraries and Components.pdf