

General Description

- The USBUART Device uses a USB interface to emulate a COM port.
- The required .inf file is automatically generated based on your component settings.
- *This is a custom component and has not been fully tested. Use at your own risk.*

Quick Start

1. To add this component to your project, simply add a dependency in your project that points to the library project containing the USBUART component. The USBUART component will then be available in your Component Catalog under 'Custom -> Communications -> USBUART'.
2. Drag a USBUART component from the Component Catalog onto your design.
3. Notice the clock errors in the Notice List window; double-click on an error to open the System Clock Editor.
4. Configure the following clocks:
 - **IMO:** Select Osc 24.000 MHz and enable the Doubler
 - **ILO:** Select 100 kHz.
 - **USB:** Enable and select IMOx2 – 48.000 MHz.
 - **Master Clock:** Select IMO – 48.000 MHz.
5. Select Build to generate APIs; refer to the Sample Firmware Source Code section for an example demonstrating the basic functionality of the USBUART component, as well as the basic setup instructions.

Input/Output Connections

None.

Parameters and Setup

Vendor ID

Each USB product must have a unique combination of Vendor ID (VID) and Product ID (PID). This 2-byte string contains the Vendor ID.

PRELIMINARY

Product ID

Each USB product must have a unique combination of Vendor ID (VID) and Product ID (PID). This 2-byte string contains the Vendor ID.

Device Release

Use this field to keep track of your current release.

Manufacturer String

This string is automatically placed in the manufacturer field within the .inf file and will show up when the device enumerates and is viewed in the Device Manager on the PC.

Product String

This string is automatically placed in the description field within the .inf file and will show up when the device enumerates and is viewed in the Device Manager on the PC.

Configure 'USBUART'

Name:

Properties Built-in

Device Attributes

Vendor ID	0x	<input type="text" value="1111"/>
Product ID	0x	<input type="text" value="2222"/>
Device Release	0x	<input type="text" value="3333"/>
Manufacturer String	<input type="text" value="Cypress"/>	
Product String	<input type="text" value="Test Product"/>	

The .inf file can be found in ...\\Generated_Source\\PSoC3

Device Driver:

This component automatically creates a basic .inf file to allow your USBUART project to connect to your PC. The .inf file will be located in ...YourProject/Generated_Source/PSoC3/. When you plug your device in for the first time, point Windows to this directory to find the .inf file and install the necessary drivers.

PRELIMINARY



Application Programming Interface

This component has all of the standard USBFS API available. Please refer to the USBFS component datasheet for details on using this functionality.

Only a small subset of the USBUART functions that were previously available on PSoC1 have been implemented in this component.

Function	Description
uint8 USBUART_bGetRxCount(void)	Get the size of valid data in the receive buffer.
void USBUART_ReadAll (uint8 *pData)	Reads all data from the receive buffer into pData.
void USBUART_Write(uint *pData, uint8 bLength)	Writes the data to the Tx Buffer.
uint8 USBUART_bTxIsReady(void)	Checks to see if the device is ready to send data. If not, 0 is returned. Otherwise the Tx buffer size is returned.
uint8 USBUART_Init(void)	Initializes the component.

Sample Code

This example will echo any characters typed into a terminal program on the PC.

```

uint8 bCount;                                /* count of data */
uint8 bIndex;
uint8 baBuffer[128];

uint32 wGetDTERate;

void main()
{
    CYGlobalIntEnable;                        /* Enable Global Interrupts */

    USBUART_Start(0, USBUART_3V_OPERATION);    /* Start USBFS Operation/device 0
    and with 3V operation */
    while(!USBUART_bGetConfiguration());        /* Wait for Device to enumerate */
    USBUART_Init();                            /* initialize the USBUART */

    while(1)
    {
        /* Get Baud Rate (not used in this example): */
        //USBUART_dwGetDTERate(&wGetDTERate);

        /* Receive Data and Echo: */
        bCount = USBUART_bGetRxCount();        /* get the USB Rx data size */
        if( bCount != 0 )
        {
            USBUART_ReadAll(baBuffer);          /* copy data to baBuffer */

            USBUART_Write( baBuffer, bCount );  /* echo / send out data */
            while(!USBUART_bTxIsReady()) {}
        }
    }
}

```

PRELIMINARY



}

}

© Cypress Semiconductor Corporation, 2010. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® Creator™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

PRELIMINARY

