CE195379 – Software Transmit UART with PSoC 3, PSoC 4 and PSoC 5LP

Objective

This code example demonstrates the usage of the Software Transmit UART Component of PSoC® Creator™.

Overview

This code example uses the Software Transmit UART Component to send test data to the terminal program on a PC.

Requirements

Tool: PSoC Creator 4.2

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22 for PSoC 4 and PSoC 5LP; DP8051 Keil 9.51 for PSoC 3)

Associated Parts: All PSoC 3, PSoC 4 and PSoC 5LP parts.

Related Hardware: CY8CKIT-001, CY8CKIT-030, CY8CKIT-040, CY8CKIT-041-40XX, CY8CKIT-041-41XX, CY8CKIT-042, CY8CKIT-042-BLE, CY8CKIT-042-BLE-A, CY8CKIT-044, CY8CKIT-046, CY8CKIT-048, CY8CKIT-149, CY8CKIT-050

Hardware Setup

This code example supports kits, shown in Table 1. By default, this example project is configured to run on the CY8CKIT-042 development kit from Cypress Semiconductor. The project can be migrated to any supported kit by changing the target device with **Device Selector** called from the project's context menu. Refer to Table 1 for the target device name of your kit.

For all supported kits, this code example uses the kit's default configuration. Refer to the kit's guide to ensure that the kit is configured correctly.

For all supported kits, the project includes control files to automatically assign pins with respect to the kit hardware connections during the project build. To change the pin assignments, override the control file selections in the Pin Editor of the Design Wide Resources by selecting the new port or pin number.

Table 1. Supported Kits, Corresponding Devices, and Pin Assignments

Development Kit	Series	Device	Pin Assignment: SW_Tx_UART
CY8CKIT-001	PSoC 3 (with CY8C38 module)	CY8C3866AXI-040	P0[5]
	PSoC 4200 (with CY8CKIT-038 module)	CY8C4245AXI-483	
	CY8C58LP (with CY8C58LP module)	CY8C5868AXI-LP035	
CY8CKIT-030	CY8C38	CY8C3866AXI-040	
CY8CKIT-040	PSoC 4000	CY8C4014LQI-422	
CY8CKIT-041-40XX	PSoC 4000S	CY8C4045AZI-S413	
CY8CKIT-041-41XX	PSoC 4100S	CY8C4146AZI-S433	
CY8CKIT-042	PSoC 4200	CY8C4245AXI-483	
CY8CKIT-042-BLE	PSoC 4200 BLE	CY8C4247LQI-BL483	P1[5]
CY8CKIT-042-BLE-A	PSoC 4200 BLE	CY8C4248LQI-BL483	P1[5]
CY8CKIT-044	PSoC 4200M	CY8C4247AZI-M485	P7[1]
CY8CKIT-046	PSoC 4200L	CY8C4248BZI-L489	P3[1]
CY8CKIT-048	PSoC Analog Coprocessor	CY8C4A45LQI-483	P0[5]
CY8CKIT-149	PSoC 4100S Plus	CY8C4147AZI-S475	P7[1]
CY8CKIT-050	CY8C58LP	CY8C5868AXI-LP035	P0[5]

Software Setup

For this code example, any terminal software can be used: HyperTerminal, Bray's Terminal, PuTTY, etc. The configuration of the terminal described in the "Operation" section.

Operation

1. Plug your kit board into your computer's USB port.

   Note: If you use the **CY8CKIT-001**, **CY8CKIT-030**, or **CY8CKIT-050** kit, connect it to the PC with an RS-232 cable and power source your kit.

2. Build the project and program it into the PSoC device. Choose **Debug** > **Program**. For more information on device programming, see the PSoC Creator Help and your kit's user guide.

3. The kits listed below require jumper wire connection:

   - **CY8CKIT-040** or **CY8CKIT-042**: connect the PSoC 4 UART Tx pin P0[5] to the KitProg's UART Rx pin P12[6] on J8 header.
   - **CY8CKIT-001**, **CY8CKIT-030**, or **CY8CKIT-050**: connect UART Tx P0[5] to Tx pin of the RS-232 header.

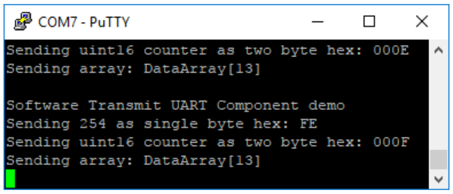
   Other supported kits do not require jumper connection. For more information, refer to the kit's user guide.

4. Run any COM terminal program. Set the terminal parameters: **Baud Rate** – 57600, **Data Bits** – 8, **Parity** – None, **Stop Bits** – 1, and connect to the corresponding port.

   Note: If you use PSoC 4 kit, open the **Device Manager** in your PC, find the device **KitProg USBUART** or **KitProg2 USBUART** under **Ports** (COM & LPT), and note the port number.

5. Observe the test messages in the Terminal window (Figure 1).

Figure 1. UART Terminal Output



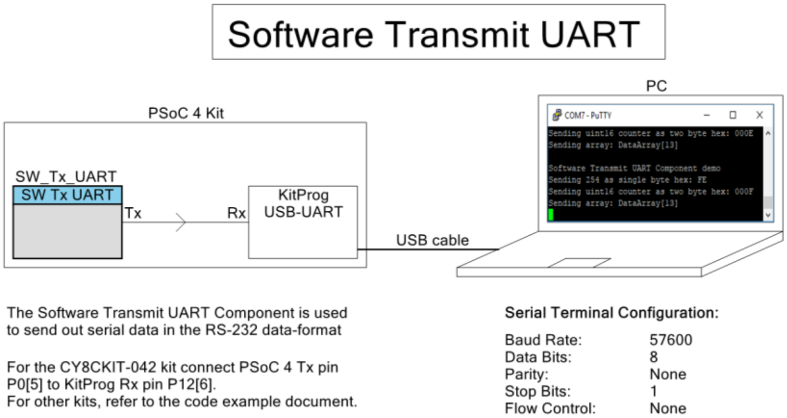
```
Sending uint16 counter as two byte hex: 000E
Sending array: DataArray[13]

Software Transmit UART Component demo
Sending 254 as single byte hex: FE
Sending uint16 counter as two byte hex: 000F
Sending array: DataArray[13]
```

Design and Implementation

The Software Transmit UART Component is used to send out serial data in the RS-232 data format. This Component uses only the Digital Output Pin as a hardware resource, so it may be used on devices without digital resources, or in projects where all digital resources are consumed. Figure 2 shows the PSoC Creator schematic for this code example.

Figure 2. Top Design Schematic



The Software Transmit UART Component is used to send out serial data in the RS-232 data-format

For the CY8CKIT-042 kit connect PSoC 4 Tx pin P0[5] to KitProg Rx pin P12[6].
For other kits, refer to the code example document.

Serial Terminal Configuration:

Baud Rate: 57600
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None

To support all kits from Table 1, selected System clock frequency of 12MHz. The maximum supported baud rate for this frequency is 57600. For more details, refer to the Software Transmit UART Component datasheet.

The firmware sets up and enables the Software Transmit UART Component, and sends test data with a 2-second interval.

Components and Settings

Table 2 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

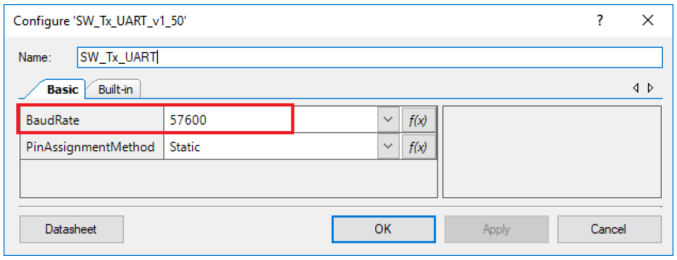
Table 2. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
Software Transmit UART	SW_Tx_UART	Sends test data to the terminal program on the PC.	See Figure 3.

For information on the hardware resources used by the Component, see the Component datasheet.

Figure 3 highlights the non-default settings for the Software Transmit UART Component.

Figure 3. Software Transmit UART Component Configuration



Configure 'SW_Tx_UART_v1_50'

Name: SW_Tx_UART

BaudRate	57600
PinAssignmentMethod	Static

Reusing This Example

This example is designed for the kits listed in Table 1. To port the design to a different PSoC 3, PSoC 4 or PSoC 5LP device and/or kit, change the target device using **Device Selector** and update the Tx pin assignment in the **Design Wide Resources Pins** settings as needed.

Related Documents

Application Notes	
AN54181 – Getting Started with PSoC 3	Introduces the PSoC 3 architecture and development tools
AN79953 – Getting Started with PSoC 4	Introduces the PSoC 4 architecture and development tools
AN77759 – Getting Started with PSoC 5LP	Introduces the PSoC 5LP architecture and development tools
PSoC Creator Component Datasheets	
Software Transmit UART	Provides software Tx UART interface.
Device Documentation	
PSoC 3 Family Datasheets	PSoC 3 Architecture Technical Reference Manual
PSoC 4 Family Datasheets	PSoC 4 Architecture Technical Reference Manual
PSoC 5LP Family Datasheets	PSoC 5LP Architecture Technical Reference Manual
Development Kit Documentation	
CY8CKIT-001 PSoC Development Kit	
CY8CKIT-030 PSoC 3 Development Kit	
CY8CKIT-050 PSoC 5LP Development Kit	

Document History

Document Title: CE195379 – Software Transmit UART with PSoC 3, PSoC 4 and PSoC 5LP

Document Number: 001-95379

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5991601	MYKZTMP1	01/12/2018	New code example

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.