Objective

This code example demonstrates the operation of the UART Component in Full Duplex mode with PSoC 3, PSoC 4, and PSoC 5LP. It also shows how to use an external interrupt (schematic interrupt) and the printf() function.

Overview

This code example project demonstrates how to communicate between the PC and the universal asynchronous receiver transmitter (UART) Component in Full Duplex mode implemented in the universal digital blocks (UDB). The UART has a receiver (RX) and a transmitter (TX). The data received by RX is looped back to TX.

This code example implements compiler-specific low-level functions for the output stream and calls the UART Component API to send data by the printf() function.

Requirements

Tool: PSoC Creator™ 3.3 SP1 or later

Programming Language: C (GCC 4.9) or later

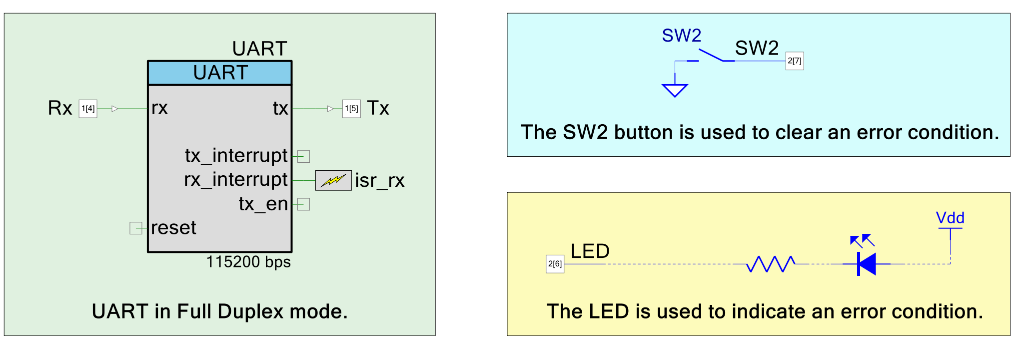
Associated Parts: PSoC 3, PSoC 4, PSoC 5LP parts with UDB.

Related Hardware: CY8CKIT-030, CY8CKIT-050, CY8CKIT-042, CY8CKIT-042-BLE, CY8CKIT-042-BLE-A, CY8CKIT-046

Design

The design uses an external interrupt (isr_rx) connected to the rx_interrupt output of the UART Component. The isr_rx interrupt reads the data received by the UART and sends it back to the PC. An interrupt is triggered when data is stored in the internal 4-byte-deep RX FIFO or when an error occurs during the receive operation. In an errorless condition, the interrupt handler passes the received data to the 4-byte-deep TX FIFO. The LED indicates an error condition. By pressing SW2, the user may clear the error condition and continue the communication. Figure 1 shows the top design schematic.

Figure 1. Top Design Schematic



Design Considerations

The project is intended to echo back an unlimited amount of data. The long-term errorless UART functionality depends on the clock frequency on both sides – the PC and device. When the clock on the PC side has a higher frequency compared to the device clock, the device receives more data than it is able to send back, and the internal 4-byte FIFO buffer gets overloaded. Use short packets to avoid such behavior or implement large software buffers.

This design can be extended by using the internal UART Component interrupts, a large internal software buffer, and the polling wraparound method in the main loop. To enable this feature, set the INTERRUPT_CODE_ENABLED define in the *common.h* file to DISABLED and increase the RX and TX buffer sizes in the advanced tab of the UART Component configuration dialog.

The printf() function formats a series of strings and numeric values and builds a string to write to the output stream. It has different implementations for different compilers. The Keil C51 compiler uses putchar(), GCC uses _write(), MDK and RVDS use fputc(), while IAR uses the __write() function to send data. This code example project has these functions implemented in the debug.c file. This enables an application to run the printf() function with any compiler.

Note: The project adds an explicit reference to the floating point printf library to allow the usage of the floating point conversion as it is not supported by the GCC compiler by default. The required code: `asm (".global _printf_float");`

The printf() function support can be disabled in the project by setting the UART_PRINTF_ENABLED define in the *common.h* file to DISABLED.

Hardware Setup

This example project is designed to run on the CY8CKIT-042-BLE development kit from Cypress Semiconductor. A full description of the kit, along with more example programs and ordering information, can be found at http://www.cypress.com/go/cy8ckit-042-BLE.

The project requires changes to configuration settings to run on other kits from Cypress Semiconductor. Table 1 lists the supported kits. To switch from CY8CKIT-042-BLE to any other kit, change the project's device with the help of the Device Selector called from the project's context menu.

Table 1. Development Kits vs Parts

Development Kit	Device
CY8CKIT-030	CY8C3866AXI-040
CY8CKIT-050	CY8C5868AXI-LP035
CY8CKIT-042	CY8C4245AXI-483
CY8CKIT-042-BLE	CY8C4247LQI-BL483
CY8CKIT-042-BLE-A	CY8C4248LQI-BL583
CY8CKIT-046	CY8C4248BZI-L489

The pin assignments for the supported kits are provided in Table 2. A control file is added to the project to control that all the pins be properly assigned after the project build.

Table 2. Pin Assignment

Pin Name	Development Kit					
	CY8CKIT-030	CY8CKIT-050	CY8CKIT-042	CY8CKIT-042-BLE	CY8CKIT-042-BLE-A	CY8CKIT-046
Rx	P0[5]	P0[5]	P0[4]	P1[4]	P1[4]	P3[0]
Tx	P0[4]	P0[4]	P0[5]	P1[5]	P1[5]	P3[1]
SW2	P6[1]	P6[1]	P0[7]	P2[7]	P2[7]	P0[7]
LED	P6[2]	P6[2]	P1[6]	P2[6]	P2[6]	P5[2]

Note: To run a code example project on the kits listed below, the pins must be connected to the headers using wires:

- **CY8CKIT-030**: connect PSoC 3 Rx pin to P5.1 (SERIAL_RX), connect PSoC 3 Tx pin to P5.2 (SERIAL_TX)
- **CY8CKIT-050**: connect PSoC 5LP Rx pin to P5.1 (SERIAL_RX), connect PSoC 5LP Tx pin to P5.2 (SERIAL_TX)
- **CY8CKIT-042**: connect PSoC 4 Rx pin to J8.10, connect PSoC 4 Tx pin to J8.9.

The define assignments required for the supported kits are in Table 3.

Table 3. Define Assignment

Define Name	Development Kit					
	CY8CKIT-030	**CY8CKIT-050**	**CY8CKIT-042**	**CY8CKIT-042-BLE**	**CY8CKIT-042-BLE-A**	**CY8CKIT-046**
LED_ON	1	1	0	0	0	0
LED_OFF	0	0	1	1	1	1

Software Setup

This example project communicates with a PC host using a UART. A HyperTerminal program is required in the PC to communicate with the kit. If you don't have a HyperTerminal program installed, download and install any serial port communication program. Freeware such as HyperTerminal, Bray's Terminal, Putty etc. is available on the web.

Follow these steps to communicate with the PC host.

1. Connect the PC and your kit with a USB cable. If you use the CY8CKIT-030 or CY8CKIT-050 kit, connect it to the PC with an RS232 cable and power source these kits.
2. If you use PSoC 4 kit, open the device manager program in your PC, find the device **KitProg USBUART** under **Ports** (COM & LPT), and note the port number.
3. Open the HyperTerminal program and select the COM port in which the kit is connected.
4. Configure the Baud rate, Parity, Stop bits, and Flow control information in the HyperTerminal configuration window. The default settings: Baud rate – 115200, Parity – None, Stop bits – 1, Flow control – None. These settings should match the configuration of the PSoC Creator UART Component in the project.
5. Start communicating with the device as explained in the Operation section.

Components / User Modules

Table 4 lists the PSoC Creator Components used in this example, and the hardware resources used by each Component.

Table 4. List of PSoC Creator Components

Component	Hardware Resources
UART	UDB, Digital clock
Rx, Tx, SW2, LED	Digital IO pins
isr_rx	Interrupt

Parameter Settings

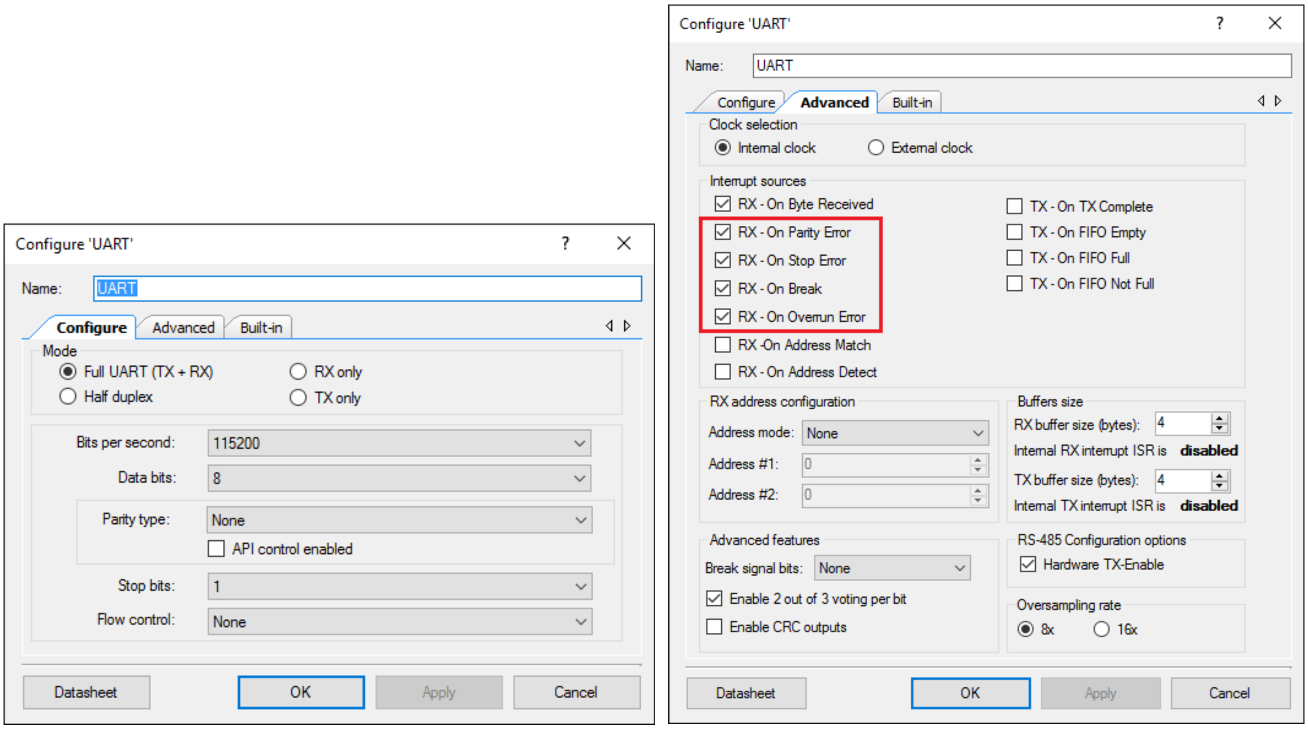
By default, the UART is configured to Baud rate – 115200, Parity – None, Stop bits – 1 and Flow control – None. These settings can be changed to match the COM port configuration on PC.

Note CY8CKIT-042-BLE and CY8CKIT-042 kits communicate through the USB-UART Bridge. Refer to the "USB-UART Bridge" section of CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit Guide for supported UART configurations.

The following interrupt sources are enabled in the Advanced tab in addition to the enabled by default RX – On Byte Received:

- RX – On Parity Error
- RX – On Stop Error
- RX – On Break
- RX – On Overrun Error

Figure 2. UART Configuration



Design-Wide Resources

The printf() function uses the dynamic memory allocation. For the proper function operation, set the Heap Size to 0x300 in the **System** tab of design-wide resource (DWR) settings.

Operation

1. Build and program the project into the development kit.
2. Run the Terminal application, press the **Reset** button on the kit and see the following lines on the Terminal window.

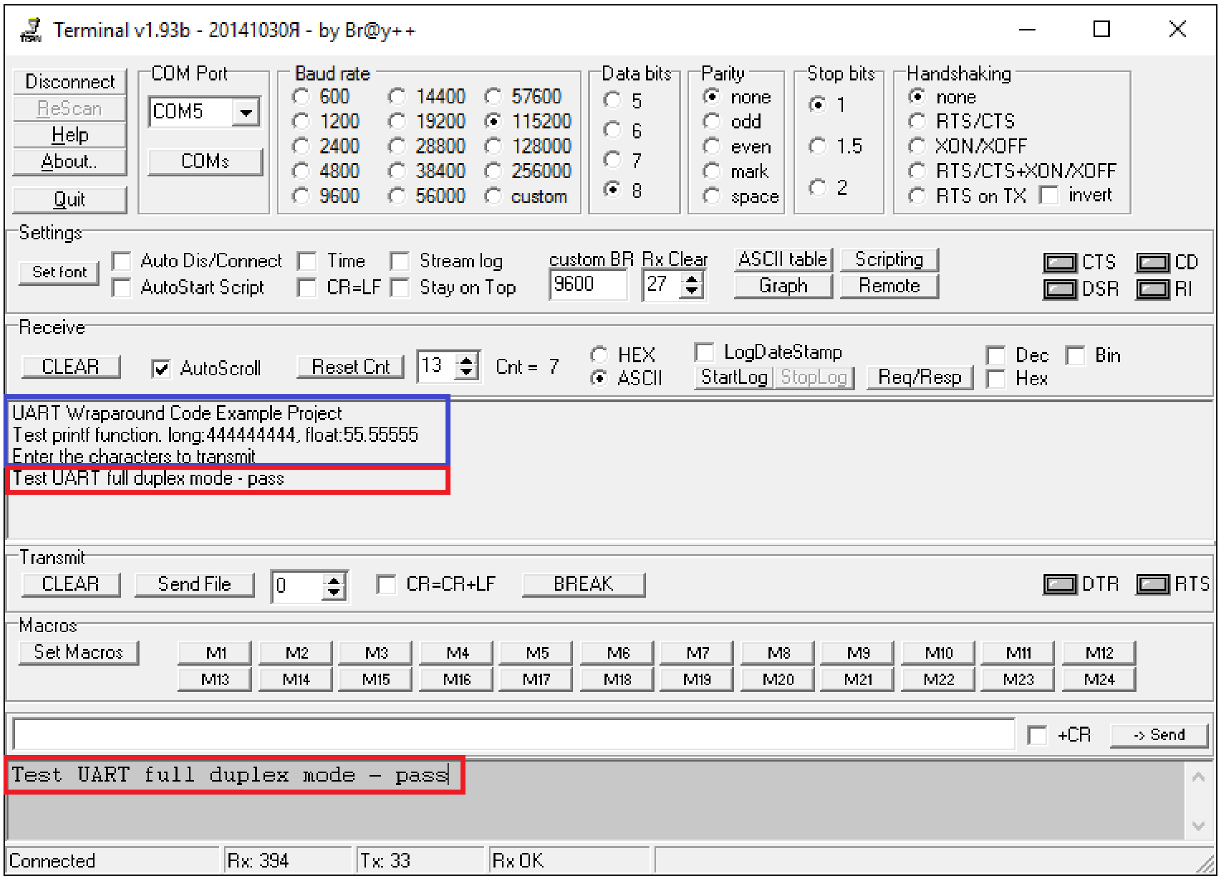
```
UART Wraparound Code Example Project
Test printf function. long:444444444, float:55.55555
Enter the characters to transmit
```

3. Start typing in the Terminal and observe the same data is received as Figure 3 shows.

   Note: The HyperTerminal can have the Local Echo setting turned ON by default, so two characters will be looped back. Make sure this parameter is turned OFF for the proper operation.

4. To verify if the project detects errors, change the **Baud rate** in the Terminal (for example to 19200) and send some data to the device. Observe that the LED is ON. Use the Debugger to check which error condition is triggered by reading the `errorStatus` global variable. Press SW2 to clear the LED indication and return the **Baud rate** configuration to 115200 to continue errorless communication.

Figure 3. Expected Results in Bray's Terminal Application



Related Documents

Table 5 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 5. Related Documents

Application Notes is		
AN79953	Getting Started with PSoC® 4	Describes PSoC 4 and shows how to build a first PSoC Creator project.
AN54181	Getting Started with PSoC® 3	Describes the PSoC 3 architecture and development environment, and shows how to create a simple design using PSoC Creator, the development tool for PSoC 3.
AN77759	Getting Started with PSoC® 5LP	Describes the PSoC 5LP architecture and development environment, and shows how to create a simple design using PSoC Creator, the development tool for PSoC 5LP.
Code Examples		
CE95389	UART Transmit with PSoC 3/4/5LP	
CE95388	UART Receive with PSoC 3/4/5LP	
CE95395	USB MIDI with PSoC 3/5LP	
PSoC Creator Component Datasheets		
UART	Universal Asynchronous Receiver Transmitter (UART)	
Interrupt	Interrupt	
Pins	Supports connection of hardware resources to physical pins	
Device Documentation		
PSoC 3 Datasheets	PSoC 3 Technical Reference Manuals	
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals	
PSoC 5LP Datasheets	PSoC 5LP Technical Reference Manuals	
Development Kit (DVK) Documentation		
CY8CKIT-030 PSoC® 3 Development Kit		
CY8CKIT-050 PSoC® 5LP Development Kit		
CY8CKIT-042 PSoC® 4 Pioneer Kit		
CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit		
CY8CKIT-042-BLE-A Bluetooth® Low Energy 4.2 Compliant Pioneer Kit		
CY8CKIT-046 PSoC® 4 L-Series Pioneer Kit		

Document History

Document Title: CE210741 UART Full Duplex and printf() Support with PSoC 3/4/5LP

Document Number: 002-10741

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5245208	NAZR	6/13/16	New spec
*A	5245203	NAZR	9/27/16	Added control file for automatic pin definition based on selected device.
*B	5739947	AESATP12	05/26/17	Updated logo and copyright.
*C	5926681	SVOZ	10/17/17	Document update, added CY8CKIT-046 and CY8CKIT-042-BLE-A support

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

Cypress Developer Community

Forums | WICED IOT 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