CE224431 – PSoC 4 UART printf

Objective

This example shows how to re-direct and use the `printf()` function for sending data, using a PSoC® Creator™ Serial Communication Block (SCB) Component configured as a UART in a PSoC 4 device. Note that the UART in this example is used only to demonstrate redirecting the `write()` function that is called by `printf()`.

Requirements

Tool: PSoC Creator 4.2

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: PSoC 4 family

Related Hardware: CY8CKIT-042 PSoC 4 Pioneer Kit

Overview

This example has one PSoC Creator project file that shows the methodology to rewrite `printf` for GCC compiler. Printf is demonstrated with a test call; it then continuously displays the number of kit button presses.

Hardware Setup

This code example is set up for CY8CKIT-042. If you are using a different kit, see Reusing This Example.

For the CY8CKIT-042, the USB-UART bridge in KitProg2 module is used. Connect the \UART:tx\ pin P0[5] to P12[6] on header J8.

Other kits use different pins for the UART. Make sure that you select the pins that are right for your kit.

Software Setup

This design requires a terminal emulator such as PuTTY or Tera Term running on your computer.

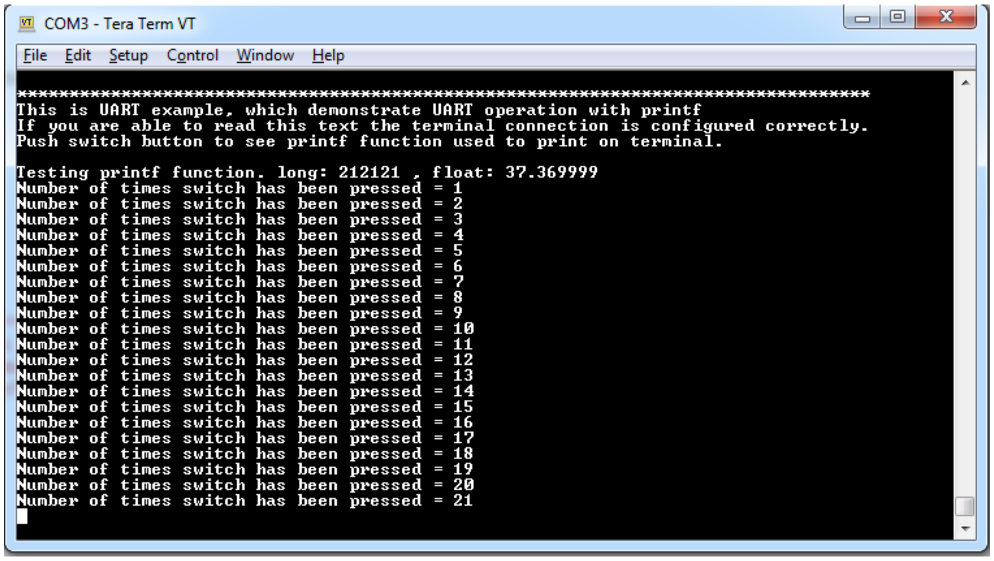
Operation

Follow these steps to communicate with the PC host:

1. Make sure you connect the correct pins, as noted in the Hardware Setup section.
2. Connect CY8CKIT-042 to your computer using a USB cable.
3. Build the project and program it into the PSoC 4 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help.
4. Open a terminal emulator on your computer and configure the program to the appropriate COM port. Configure the baud rate to 115200, data bits to 8, no parity bits, stop bit as 1, and no control flow.

5. Press the reset button on the kit and observe the welcome message printed on the terminal program, as shown in Figure 1.

Figure 1. Message Printed on the Terminal



```
********************************************************************************
This is UART example, which demonstrate UART operation with printf
If you are able to read this text the terminal connection is configured correctly.
Push switch button to see printf function used to print on terminal.

Testing printf function. long: 212121 , float: 37.369999
Number of times switch has been pressed = 1
Number of times switch has been pressed = 2
Number of times switch has been pressed = 3
Number of times switch has been pressed = 4
Number of times switch has been pressed = 5
Number of times switch has been pressed = 6
Number of times switch has been pressed = 7
Number of times switch has been pressed = 8
Number of times switch has been pressed = 9
Number of times switch has been pressed = 10
Number of times switch has been pressed = 11
Number of times switch has been pressed = 12
Number of times switch has been pressed = 13
Number of times switch has been pressed = 14
Number of times switch has been pressed = 15
Number of times switch has been pressed = 16
Number of times switch has been pressed = 17
Number of times switch has been pressed = 18
Number of times switch has been pressed = 19
Number of times switch has been pressed = 20
Number of times switch has been pressed = 21
```

6. Press kit button SW2 and observe that the terminal program displays a new line each time the button is pressed.

Design and Implementation

In this example, the SCB Component is configured as a UART. The UART first transmits a welcome message through a terminal emulator, and then sends a test `printf` call. The main program constantly checks if the flag for an interrupt occurring is HIGH (when 'switch_pin' is HIGH). If so, the program displays the number of times the button SW2 has been pressed (count is incremented on each button press). The `printf` function calls `_write()`.

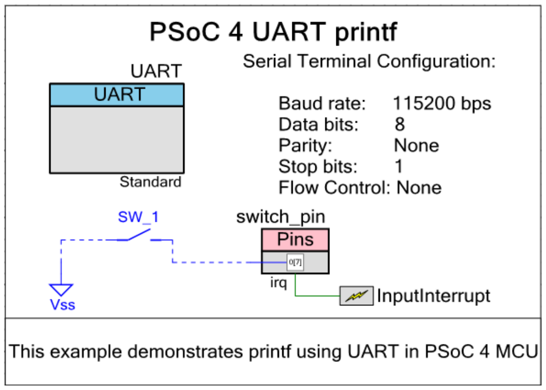
`_write` is a system call that takes in the location to write, the data to write, and the length of the data. When `printf` is called, it uses the `_write` system call to write the data to a location. This example overrides the `_write()` function for the GCC Compiler and replaces a call to the standard `putchar()` function with `UART_UartPutChar()`.

Note: The switch is active LOW, so 'switch_pin' drive mode is set to resistive pull up.

Note: The Heap Size should be at least 0x0300 bytes in your System Configuration [under Design Wide Resources] to allow for the use of floating point numbers.

The top-level design of the PSoC Creator project is shown in Figure 2.

Figure 2. UART_printf Top Design Schematic



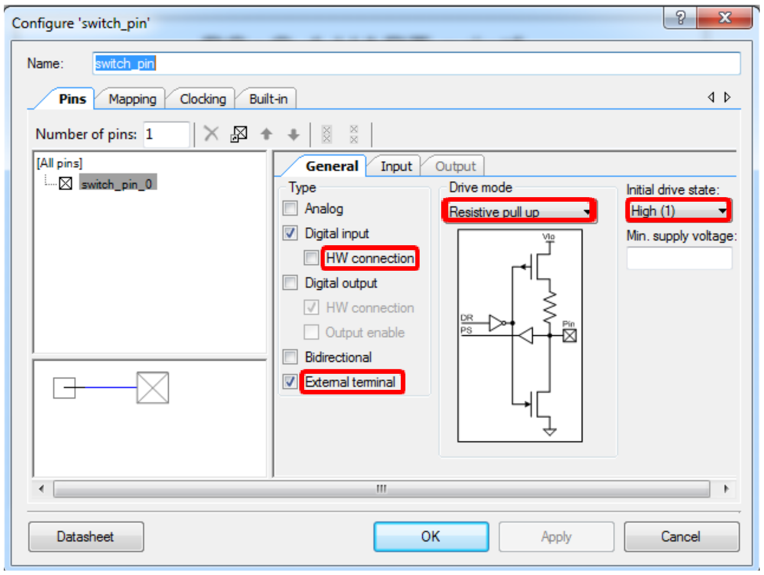
Components and Settings

Table 1 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 1. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
UART (SCB Mode)	UART	Handle UART serial communication	None
Digital Input Pin	switch_pin	Connects to button SW2 on kit board	See Figure 3.
Interrupt	InputInterrupt	Handles interrupt from button press	None

Figure 3. switch_pin Parameter Settings


Configure 'switch_pin'

Name: switch_pin

Pins | Mapping | Clocking | Built-in

Number of pins: 1

[All pins]
switch_pin_0

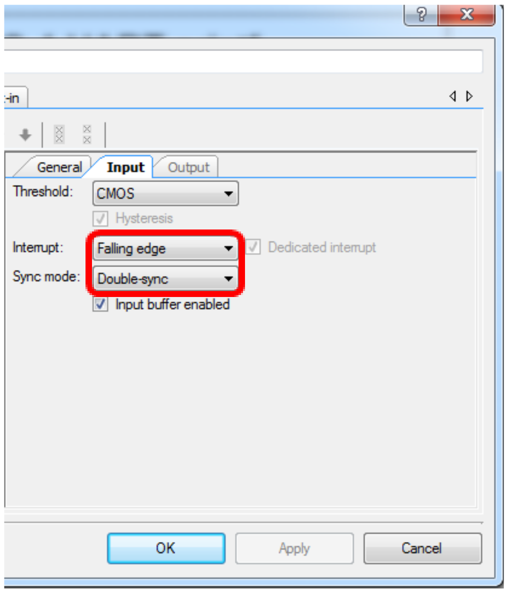
General | Input | Output

Type
- [ ] Analog
- [✓] Digital input
  - [ ] HW connection
- [ ] Digital output
  - [✓] HW connection
  - [ ] Output enable
- [ ] Bidirectional
- [✓] External terminal

Drive mode: Resistive pull up

Initial drive state: High (1)

Min. supply voltage:


General | Input | Output

Threshold: CMOS

- [✓] Hysteresis

Interrupt: Falling edge
- [✓] Dedicated interrupt

Sync mode: Double-sync

- [✓] Input buffer enabled

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

Reusing This Example

This example is designed for the CY8CKIT-042 pioneer kit. To port this design to a different PSoC 4 device, kit, or both, do the following:

1. In PSoC Creator, select **Project** > **Device Selector** to change the target device. Select your device as listed in Table 2.
2. Make sure that the **SysClk Desired Frequency** is set to 24 MHz after the device is changed.
3. In PSoC Creator Workspace Explorer, select the **Clocks** interface listed under **Design Wide Resources**.
4. Set the **SysClk Desired Frequency** to 24 MHz, if it is not already.
5. Update the pin assignments in the Design Wide Resources Pins settings as needed (see Table 3).

Table 2. Development Kits and Associated Devices

Development Kit	Device
CY8CKIT-041	CY8C4146AZI-S433
CY8CKIT-042	CY8C4245AXI-483
CY8CKIT-042-BLE	CY8C4247LQI-BL483
CY8CKIT-044	CY8C4247AZI-M485
CY8CKIT-046	CY8C4248BZI-L489
CY8CKIT-048	CY8C4445AZI-483

Table 3. Pin Assignments for Different Kits

Pin Name	Development Kit					
	CY8CKIT-041	CY8CKIT-042	CY8CKIT-042-BLE	CY8CKIT-044	CY8CKIT-046	CY8CKIT-048
\UART:tx\	P0[5]	P0[5]	P1[5]	P7[1]	P3[1]	P0[5]
switch_pin	P0[7]	P0[7]	P2[7]	P0[7]	P0[7]	P0[3]

For the CY8CKIT-048, connect \UART:tx\ to P12[6] on header J16. All other listed devices connect through header J8.

In some cases, a resource used by a code example (for example, a Universal Digital block) is not supported on another device. In that case the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on what a r device supports.

Related Documents

Application Notes	
AN79953 Getting Started with PSoC 4	Describes PSoC 4 and shows how to build the attached code example
Code Examples	
CE224406 PSoC 4 UART	This code example shows how to use the PSoC Creator Serial Communication Block (SCB) Component configured as a UART in a PSoC 4 device.
PSoC Creator Component Datasheets	
SCB	A multifunction hardware block that implements the following communication components: I2C, SPI, UART, and EZI2C
DMA	Transfer data to and from memory, components, and registers that is independent of the CPU.
Device Documentation	
PSoC 4 Datasheets	PSoC 4 Technical Reference Manuals
Development Kit (DVK) Documentation	
CY8CKIT-042 PSoC® 4 Pioneer Kit	
CY8CKIT-044 PSoC® 4 M-Series Pioneer Kit	
PSoC 4 Kits	
Tool Documentation	
PSoC Creator	Go to the downloads tab for Quick Start and User Guides

Document History

Document Title: CE224431 - PSoC 4 UART printf

Document Number: 002-24431

Revision	ECN	Submission Date	Description of Change
**	6260221	07/28/2018	New code example
*A	6802764	02/25/2020	Fixed broken links. Updated to current code example template. Minor updates to document content.

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 | Code Examples | 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