Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE219765 – PSoC 6 MCU Event Profiling

Objective

This example demonstrates the PSoC® 6 MCU profiler block event monitoring, using UART communications as the activity. The profiler provides insight into the device to identify activities that may affect energy consumption in low-power systems.

Overview

This example uses PSoC 6 MCU profiler block to observe the behavior of the SCB in UART mode. It demonstrates how to use the profiler to see the impact that different programming techniques can have on performance. In this case, it shows the difference between interrupt-driven code and code that polls the UART driver.

Requirements

Tool: PSoC Creator™ 4.2 with Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (ARM® GCC 5.4 -2016-q2-update; MDK ARM Compiler 5.06 update 3 (build 300)

Associated Parts: All PSoC 6 MCU devices

Related Hardware: CY8CKIT-062 BLE

Design

This example demonstrates how to set up and use the profiler driver. This includes enabling the profile block, configuring the profile driver, selecting the source signal to monitor, and setting the input clock signals. The example starts and stops the profile count around a simple UART character output routine. In this example, the profile driver counts events (the number of UART accesses), rather than clock cycles. The code repeats the test twice, once using interrupts, and again using polling of the UART status.

Each test sends a string multiple times. The string grows by one character each time the string is sent. The profiler counts the number of times any of the UART registers are accessed. These include the TX data, RX data, and Status registers.

The design communicates the results via UART to a terminal window. The design displays an incrementing string and real-time profiler results concurrently.

Configuration of the three GPIO used for driving the RGB LED is in *main_cm4.c* lines 127–134. These are easily changed or maybe deleted without affecting the overall demo. The LED blinks red for the interrupt test, blue for the polling test, and green on completion.

Design Considerations

This code example runs on CY8CKIT-062-BLE, which has a PSoC 6 MCU device. To port the design to other PSoC devices and kits, change the target device using PSoC Creator **Project** > **Device Selector**, and pin assignments in the Design Wide Resources window. The profile module does not require special setup from other modules or clocks. The entire code example can be easily moved to any other platform that implements the profiler in hardware.

The only non-default setting in this example is to slow down the UART from the default 115200 baud to 9600 baud. This makes the output display more interesting. The baud rate can be adjusted as long as you ensure that the UART and the terminal window use the same baud rate. Setting a fast baud rate (>19200 baud) causes the display to update so quickly that one cannot see the updating as characters are pushed to the screen during the test. The UART in this example is using the SCB5 hardware and pinout. Running on different hardware might require changing the SCB or pin placement.

Each counter in the profile module is a 32-bit register in hardware. Depending on circumstances, this may not be sufficient. When counting clock cycles with a high-frequency clock source, a counter can overflow in less than 30 seconds. The Peripheral Driver Library (PDL) profiler driver implements an overflow-handling mechanism in software. The default interrupt routine (provided in the PDL) increments an additional 32-bit counter (in software) each time a profiling counter overflows. This creates an effective 64-bit counter, automatically handling overflows, and returns 64-bit results (4,000 year overflow). The profiler IRQ can be redirected as desired to customize behavior. The example enables the overflow interrupt, although there is in fact no overflow in this example.

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly. Use of Tera Term or PuTTY terminal application on the PC will accept the ANSI ESC sequences used to format the display. Terminal communication is set to 9600, no parity, and one stop bit.

Software Setup

The example uses the newlib-nano library for access to routines that handle 64-bit parameter outputs through the `printf()` function. The newlib library must be used to print double double or long long parameters correctly. The option is set from **Project** > **Build Settings** > **CM4 ARM GCC 5.4** > **Linker** > **General**. To enable the full newlib library for Cortex-M4, set the Linker option Use newlib-nano to false.

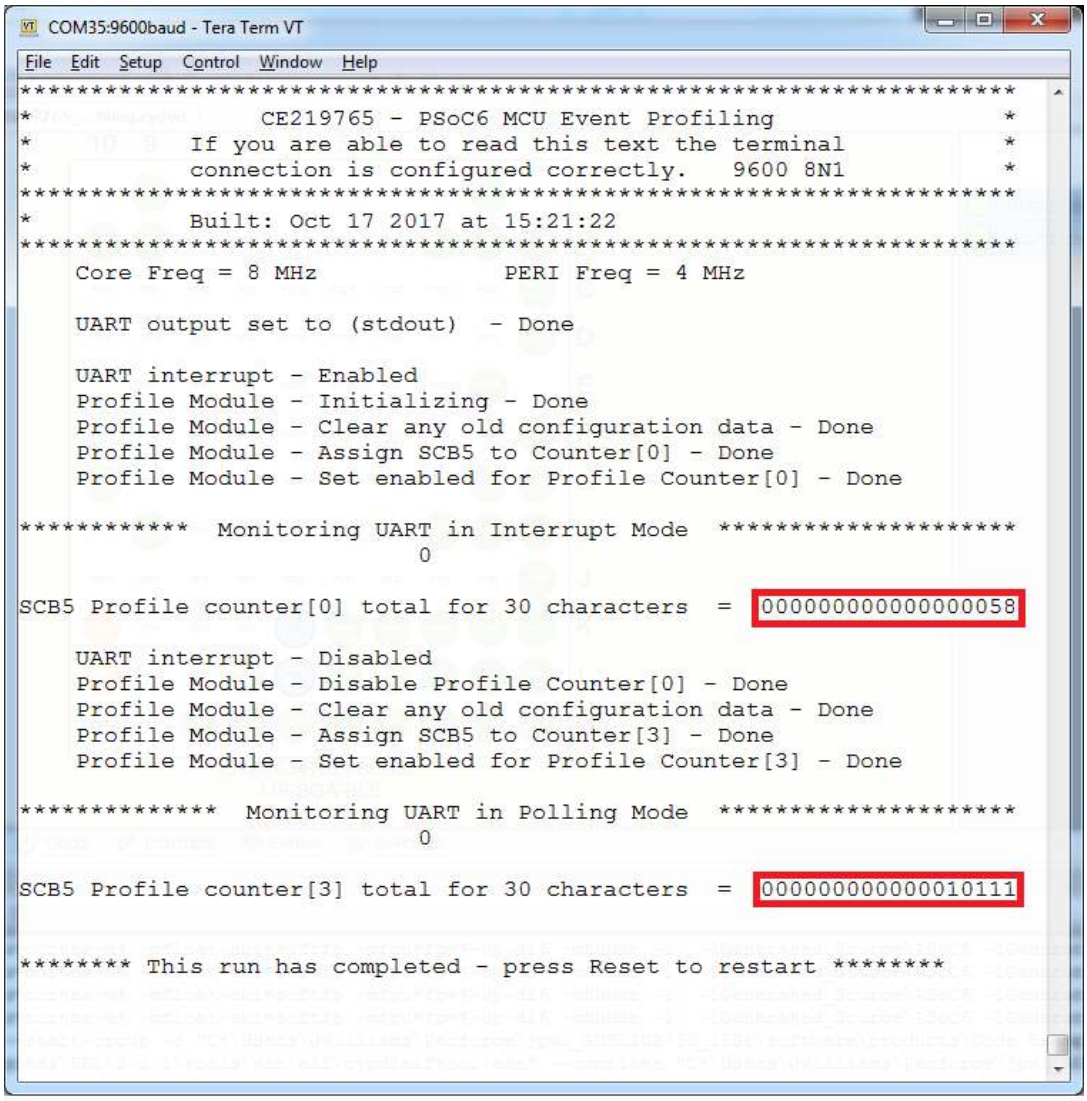
If you increase the clock speed, the number of counts on the polling side increases as well, because the core can execute the TX empty checks many more times while polling

The `myTestNumb` parameter determines how many times the test string is sent, and how long the longest string is. The value is limited to maximum 60, or an overflow will occur as the output string `myString[]` is limited to 60 characters.

Operation

1. Plug the CY8CKIT-062 kit board into your computer's USB port.
2. Open a terminal window and connect to the COM port used by the board. Configure the serial port to 8N1 at 9600 baud.
3. Build the project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation. The code begins execution automatically.
4. Observe the results in the terminal window. Figure 1 shows output in the terminal window.

Figure 1. Terminal Window Displays Results



The results from the interrupt-driven and polling modes are highlighted. The polling count represents all the times the core checked the serial output data-ready bit as it waits to place the next character into the queue.

If you want more control and to step through the code, use the debugger and launch the code with the RUN button.

Components

Table 1 lists the PSoC Creator Components used in this example, the hardware resources used by each, and the non-default parameters.

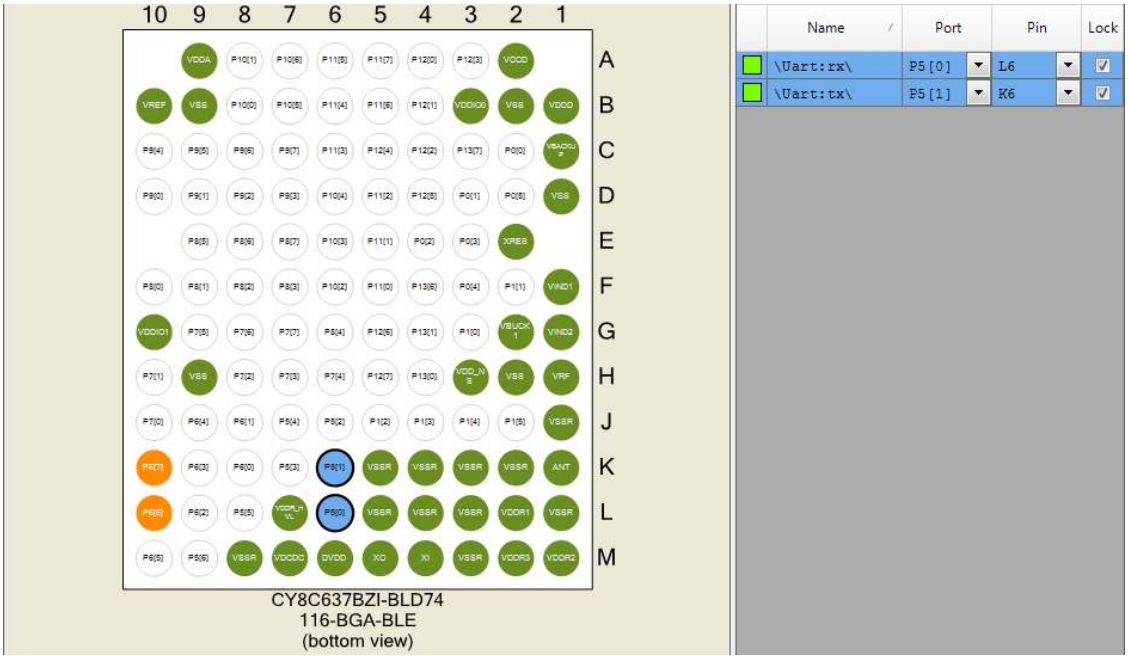
Table 1. PSoC Creator Components

Instance Name	Hardware Resources	Parameters
UART	1 SCB (5) 1 Interrupt [CM4]	Baud Rate (bps) = 9600

Design-Wide Resources

The example uses default clock settings. Figure 2 shows the pin allocation for CY8CKIT-062 BLE. Reassign pins if you port to other hardware.

Figure 2. Pin Allocation



Software assigns the RGB LED control to the following pins configured as output. Red = P0.3, Green = P1.1, Blue = P11.1

Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Introduction to PSoC 6 MCU with BLE connectivity.
PSoC Creator Component Datasheets		
UART	Provides UART settings. Review the PDL API Reference documentation for the driver.	
Peripheral Driver Library Documentation		
Energy Profiler	Review the PDL API Reference documentation for the driver.	
Device Documentation		
PSoC 6 BLE Datasheets	PSoC 6 BLE Technical Reference Manual	
Development Kit (DVK) Documentation		
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit		

Document History

Document Title: CE219765 – PSoC 6 MCU Event Profiling

Document Number: 002-19765

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5884262	JPWI	09/19/2017	New code example
*A	5932714	JPWI	10/17/2017	Modified title-removing SCB. Fixed quality of figure 1

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

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