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.

www.infineon.com

Objective

These examples demonstrate the two CPU cores in PSoC 6 MCU doing separate independent tasks, and communicating with each other using shared memory and the inter-processor communication (IPC) block.

Overview

The first example shows the two CPUs in PSoC 6 MCU – ARM® Cortex®-M0+ (CM0) and ARM Cortex-M4 (CM4) – doing independent tasks. The tasks are simple; each task blinks a separate LED using a firmware delay.

The second example uses the inter-processor communication (IPC) block in PSoC 6. Using the IPC, the CPUs share a portion of SRAM and communicate in a simple mutex/semaphore-based application.

Requirements

Tool: PSoC Creator™ 4.2 with 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 parts with dual cores

Related Hardware: CY8CKIT-062-BLE

Design

There are two examples:

1. **Basic Example:** The two CPUs each blink a separate LED. The hardware design uses only device pins for blinking LEDs, as Figure 1 shows.

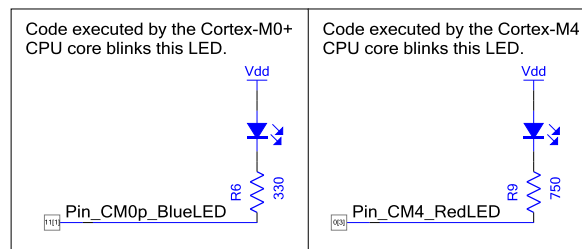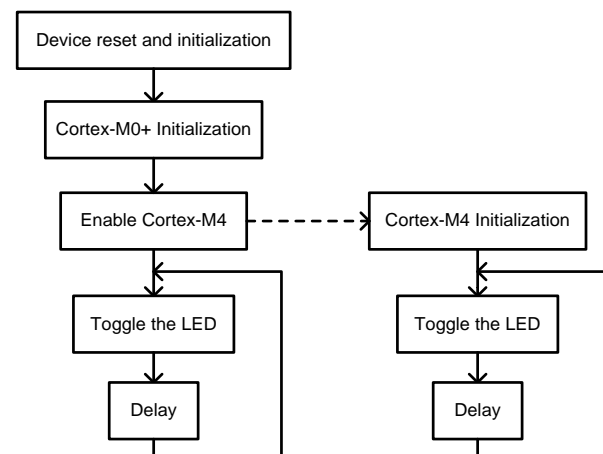Figure 1. PSoC Creator Project Schematic for Dual CPU Cores Controlling Separate Components

Figure 2 shows the firmware design. Note that after PSoC 6 device reset, CM0 always executes first while CM4 is held in a reset state. PSoC Creator automatically generates code in main() for CM0 (file *main_cm0p.c*) to enable CM4.

Figure 2. Dual CPU Basic Flowchart

2. **Inter-Processor Communication (IPC) Example:** The CPUs communicate with each other using a mutex to control access to a shared variable. Each CPU performs actions based on the value in the variable. The firmware uses the IPC application programming interface (API) in the Peripheral Driver Library (PDL). There is no PSoC Creator Component support for the IPC; nothing need be placed onto the PSoC Creator project schematic.

Figure 3 shows the firmware design. The CPUs update a one-byte shared variable in SRAM as follows:

□ CM0 increments the least-significant (LS) 4-bit nibble, then waits for the most-significant (MS) nibble to equal the LS nibble.

□ CM4 waits for the LS nibble to not equal the MS nibble, then copies the LS nibble into the MS nibble to make the two halves of the byte equal.

Figure 3. Dual CPU Shared Memory Communications Flowchart
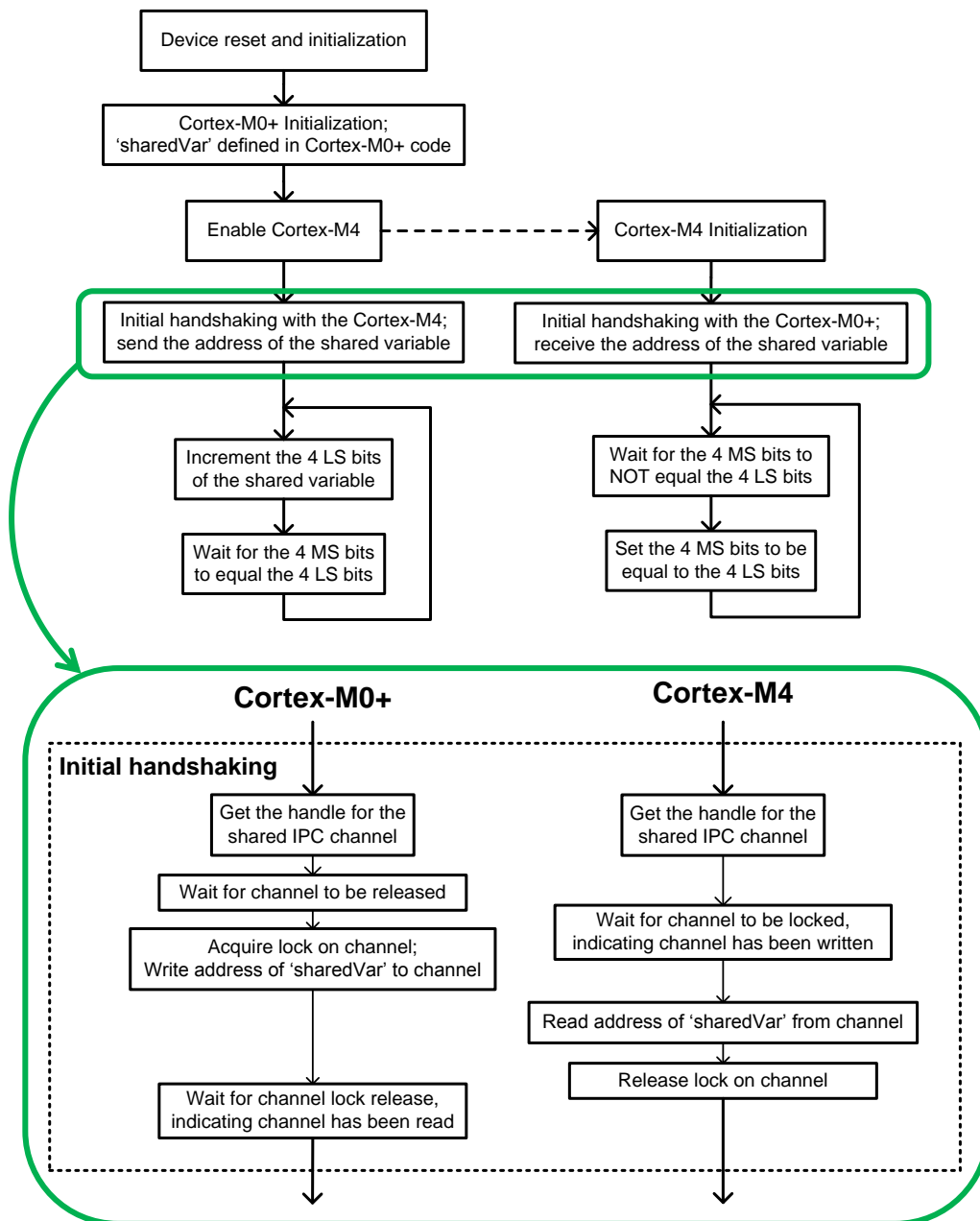
In the main() functions for both CPUs, the shared variable is never accessed directly. Instead, utility functions provide mutex lock and release access for reading and writing the variable. These utility functions in turn call PDL IPC driver functions that provide an atomic IPC channel lock and release capability. This ensures that only one CPU at a time accesses the shared variable.

The utility functions are in the code example project files *ce216795_common.h/.c*. They are linked into the executables for both CPUs, and thus they are effectively executed simultaneously by both CPUs.

Four pins are included for debug purposes, as Figure 4 shows:

Figure 4. PSoC Creator Project Schematic for Dual CPU Cores Sharing Memory

Code executed by the Cortex-M0+ CPU core uses these pins.	Code executed by the Cortex-M4 CPU core uses these pins.
[I1] Pin_CM0p_Reading	[I3] Pin_CM4_Reading
[I2] Pin_CM0p_Writing	[I4] Pin_CM4_Writing

Design Considerations

Basic Example

CY8CKIT-062-BLE has one RGB LED module. Therefore, in the LED blink example, one of four colors may be displayed at any time: black (both LEDs OFF), blue, red, and purple (both LEDs ON). To see the transitions, keep the two blink rates low, and different from each other.

If the pins are on the same GPIO port, only the following GPIO API functions should be used to update the pins: `GPIO_Write()`, `GPIO_Set()`, `GPIO_Clr()`, and `GPIO_Inv()`. For more information, see the PSoC Creator Pins Component datasheet or PDL documentation.

This code example is easily portable to the CY8CKIT-062. This kit has the same pin assignments for the LEDs, button, and communication channels as CY8CKIT-062-BLE. Change the device to CY8C6247BZI-D54.

Inter-Processor Communication (IPC) Example

The example may be switched such that the shared variable is defined in the CM4 code and its address sent to CM0.

The example includes error handling in the form of a `HandleError()` function, which is called if an IPC error or a mutex lock/release timeout is detected. The function just drops into a placeholder loop; it can be modified for application-specific error handling.

A number of other IPC-based code examples are available; they demonstrate more complex features of the IPC block and PDL driver. For more information, see Related Documents.

Dedicated and Shared Resources

This code example shows two general ways to allocate resources (e.g., pins, UARTs) to two CPUs:

- **Dedicate a resource to a CPU.** A good practice is to show on the project schematic the CPU that "owns" the resource, as Figure 1 shows. When a project is built, a separate executable is built for each CPU. Include code to use the resource only in the firmware for the desired CPU.

- **Share memory or other resources between the CPUs.** The IPC shared memory example shows how a mutex may be implemented to share memory between the CPUs. Use the same technique to share a resource such as a UART.

Flash and SRAM memory that is allocated in a CPU's executable is generally separate from that for the other CPU. If custom sections and section placement are defined in the CPUs' linker scripts, you must ensure that the sections do not overlap. Conversely, another way to share memory is to define custom sections that have the same address.

Operation

Plug the CY8CKIT-062-BLE kit board into your computer's USB port.

LED Blink Example

1. Build the project and program it into the PSoC 6 device. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.

2. Confirm that the kit's blue and red LEDs blink.

IPC Shared Memory Example

1. Build the project and program it into the PSoC 6 device. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.

2. Connect oscilloscope probes to the debug pins as needed. Confirm that the oscilloscope display is similar to Figure 5, when all four pins are monitored.

Figure 5. Screenshot Showing Dual CPUs Reading and Writing Shared Memory
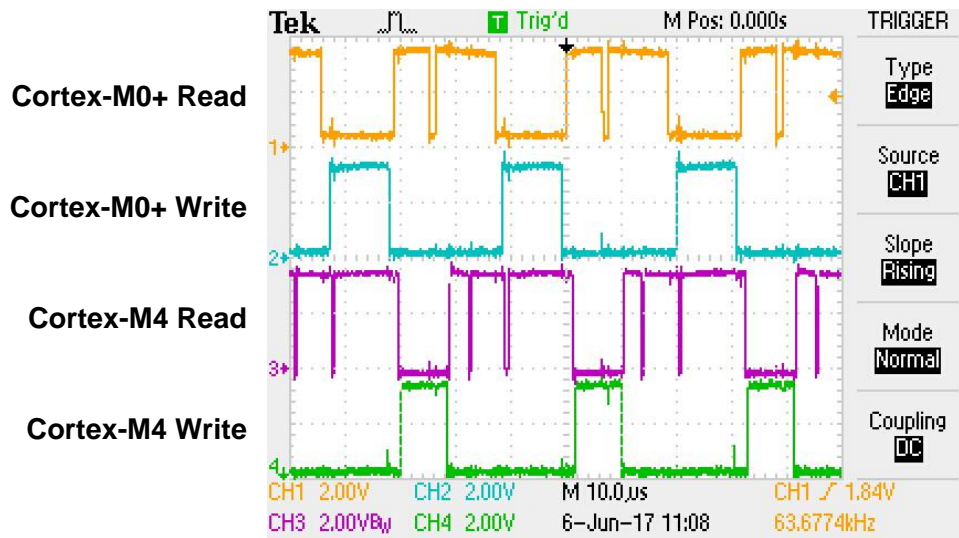
Components

Table 1 lists the PSoC Creator Components used in this example, as well as the hardware resources used by each.

Parameter Settings

In the configuration dialog for each Pin Component, the "HW Connection" box is unchecked because the pin is controlled only by the CPU.

Table 1. PSoC Creator Components

Component	Instance Name	Hardware Resources
Pin	Pin_CM0_BlueLED Pin_CM4_RedLED	2 pins
Pin	Pin_CM0_Reading Pin_CM0_Writing Pin_CM4_Reading Pin_CM4_Writing	4 pins

Related Documents

Table 2 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component / user module datasheets.

Table 2. Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 BLE	Describes the PSoC 6 BLE, and how to build a basic code example.
AN215656	PSoC 6 Dual-Core MCU System Design	Presents theory and design considerations related to this code example.
Code Examples		
TBD	TBD	Other IPC code examples
PSoC Creator Component Datasheets		
Pins	Supports connection of hardware resources to physical pins	
Device Documentation		
PSoC 6 Datasheets	PSoC 6 Technical Reference Manuals	
Development Kit (DVK) Documentation		
PSoC 6 Kits		

Document History

Document Title: CE216795 - PSoC® 6 MCU Dual-Core Basics

Document Number: 002-16795

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5583276	MKEA	01/17/2017	New code example
*A	5642351	MKEA	03/07/2017	Project update Updated template
*B	5714581	MKEA	06/09/2017	Updated document and project for release versions of PSoC Creator 4.1 and PDL 3.0.0. Changed the names of the code example common source files. Corrected an error in Figure 1.
*C	5853449	MKEA	08/04/2017	Updated document and project for build versions of PSoC Creator 4.2 and PDL 3.0.1. Added support for MDK compiler. Updated some links to other documents. Ported to new code example document template. Confidential tag removed.
*D	5890605	MKEA	09/20/2017	Updated shared memory project to align with a PDL IPC driver change.

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.