CE222793 – PSoC 6 MCU: Motion Sensor (RTOS)

Objective

This code example demonstrates how to interface a PSoC® 6 MCU with a BMI160 motion sensor. This example reads the steps counted by the sensor to emulate a pedometer. Raw motion data is also read and used to estimate the orientation of the board.

Overview

This example configures and reads data from a BMI160 motion sensor using PSoC 6 MCU. The example uses the BMI160 motion sensor to detect and count steps from activities such as walking or running, emulating the functionality of a pedometer. The motion sensor's accelerometer data is also read and converted to indicate the orientation of the sensor with respect to the ground. The step count and orientation information is displayed on the E-INK display.

This code example assumes that you are familiar with the PSoC 6 MCU device and the PSoC Creator™ Integrated Design Environment (IDE). If you are new to PSoC 6 MCU, see the application note AN210781 - Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity.

This code example uses FreeRTOS. See PSoC 6 101: Lesson 1-4 FreeRTOS training video to learn how to create a PSoC 6 FreeRTOS project with PSoC® Creator™. Visit the FreeRTOS website for documentation and API references of FreeRTOS.

Requirements

Tool: PSoC Creator 4.2; Peripheral Driver Library (PDL) 3.0.3

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22)

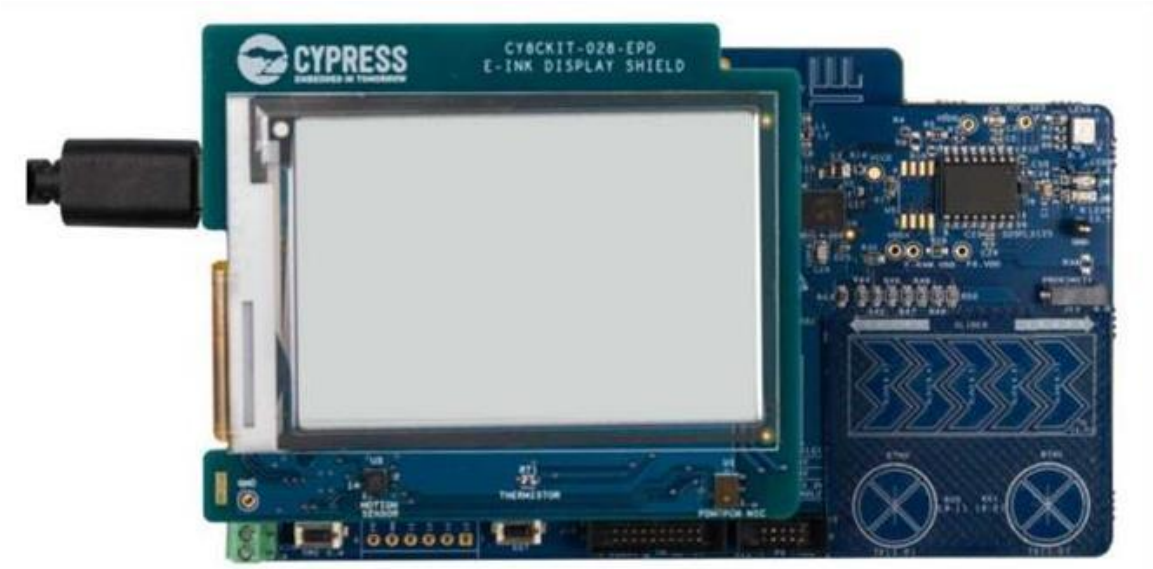
Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

Plug in the E-INK display shield on to the Pioneer board as Figure 1 shows.

Figure 1. Hardware Setup



Set the switches and jumpers as shown in Table 1.

Table 1. Switch and Jumper Selection

Switch / Jumper	Position	Location
SW5	3.3 V	Front
SW6	PSoC 6 BLE	Back
SW7	V_{DDD}/KitProg2	Back
J8	Installed	Back

Note: This code example does not support supply voltages other than 3.3 V due to limitations on the voltage required for the inertial measurement unit (IMU) and RGB LED.

Software Setup

Install the CY8CKIT-62-BLE PSoC 6 BLE Pioneer Kit software, which contains the required software to evaluate this code example. No additional software setup is required.

Operation

1. Connect the Pioneer Board to your PC using the provided USB cable through the USB connector (J10).
2. Open a terminal software such as Tera Term and select the KitProg2's COM port with a baud rate setting of 115200 bps. Use the default settings for other serial port parameters.
3. Build the project and program it into the PSoC 6 MCU. Choose **Debug** > **Program**. For more information on device programming, see the CY8CKIT-062-BLE kit guide. Flash for both CPUs is programmed in a single program operation.

Note: During the build process, do not replace *stdio_user.h* and *FreeRTOSConfig.h* file if prompted by PSoC Creator.

On successful programming, the program loads and refreshes the E-INK display as shown in Figure 2. Confirm that the terminal application displays the code example title and the initial step counts as shown in Figure 3.

Figure 2. E-INK Display on Program Startup

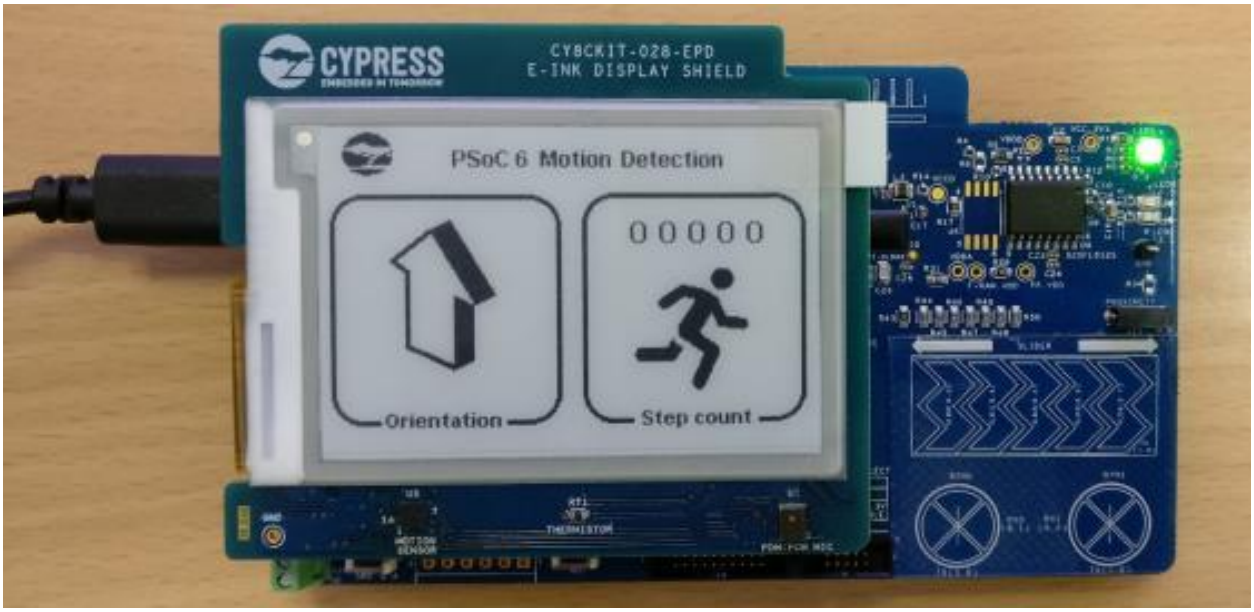
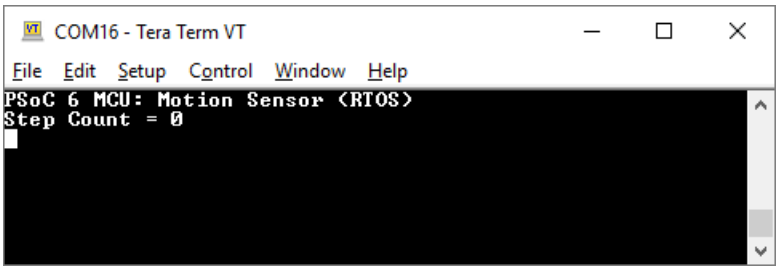


Figure 3. Terminal Application Displaying Startup Message



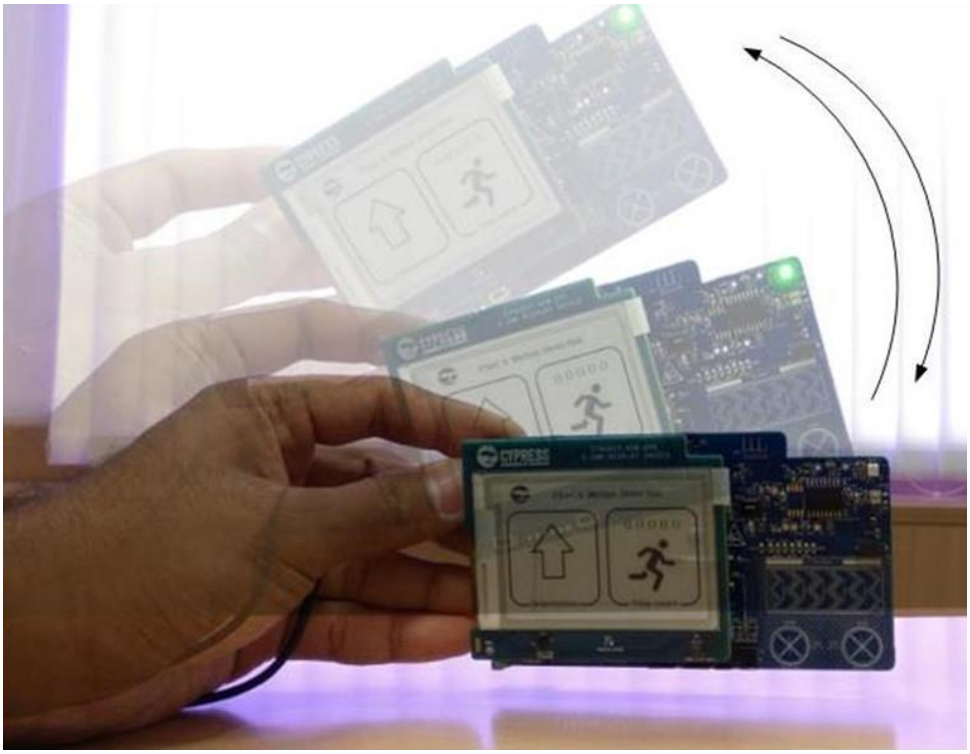
Note: If the terminal displays an error message, check the connection of the Motion Sensor or E-INK Display shield with the Pioneer Baseboard.

4. To test the step counter, hold the kit in your hand and rock it back and forth to simulate walking motion as shown in Figure 4. Do not shake the board vigorously because this motion pattern might be rejected by the motion sensor's internal filter.

   When a valid motion pattern is detected, the sensor provides a step detector interrupt signal. Notice that the green LED toggles with each step detected.

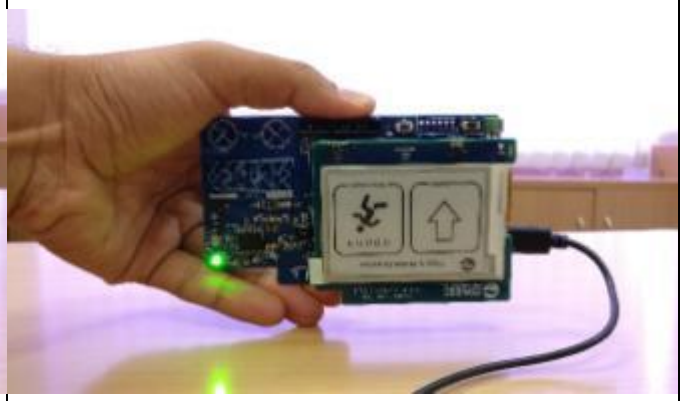
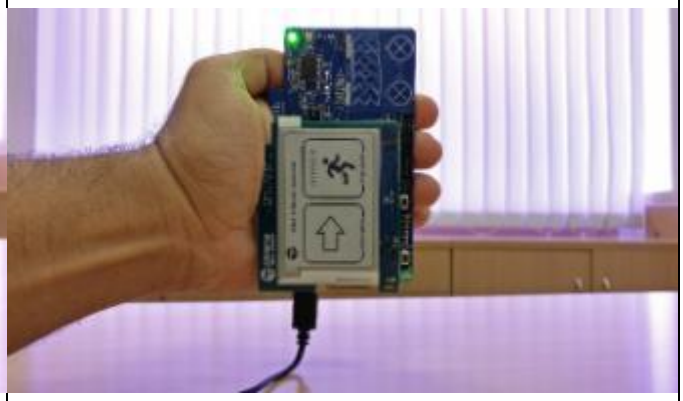
   When starting to count steps from an idle state, the sensor starts incrementing the step counter internally, but waits for a few consecutive steps (typically 4) to be registered before reporting the count. Henceforth, consecutive steps are reported as they are detected. When steps are no longer detected and motion sensor is idle, this process is repeated and step counting is resumed.

Figure 4. Simulating Step Motion



5. The accelerometer sensor data is used to estimate the board's orientation. The E-INK display shows an "arrow" graphic indicating one of six orientation states as shown in Table 2. Changing the orientation of the board between portrait and landscape orientations updates the arrow graphic to keep it pointing towards the ceiling. When the board is placed so that the display is parallel to the ground (face up/face down), a 3-D arrow pointing away from or into the board is displayed.

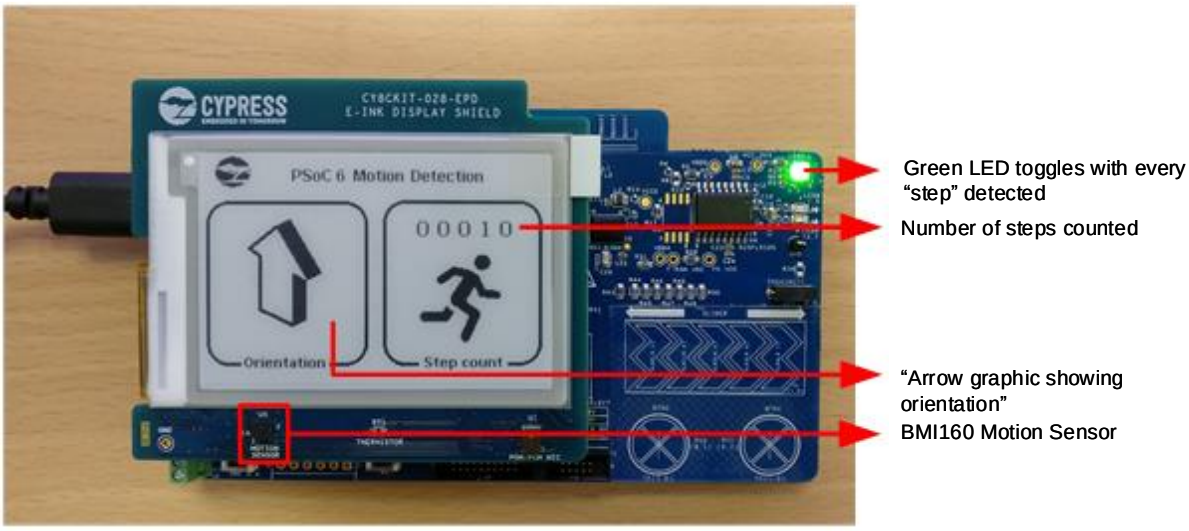
Table 2. Orientation States

	
ORIENTATION_TOP_EDGE = 0 Top edge of the board points towards the ceiling	ORIENTATION_BOTTOM_EDGE = 1 Bottom edge of the board points towards the ceiling
	
ORIENTATION_LEFT_EDGE = 2 Left edge of the board points towards the ceiling	ORIENTATION_RIGHT_EDGE = 3 Right edge of the board points towards the ceiling
	
ORIENTATION_DISP_UP = 4 Display faces up (towards the sky/ceiling)	ORIENTATION_DISP_DOWN = 5 Display faces down (towards the ground)

Design and Implementation

The E-INK Display Shield (CY8CKIT-028-EPD) contains a BMI160 motion sensor (U5), which is a low-power inertial measurement unit (IMU) providing 3-axis acceleration and 3-axis gyroscopic measurements. PSoC 6 MCU interfaces to this sensor and reads motion data, which is converted into two outputs: Orientation and Step count, and displayed on the E-INK display.

Figure 5. E-INK Display Showing Motion Sensor Outputs



Note: E-INK displays consume no power to retain the display. For more details on E-INK display, see the code example CE218133 - PSoC 6 MCU E-INK Display with CapSense.

The BMI160 motion sensor is interfaced to PSoC 6 MCU using an I^2C interface and two interrupt pins. BMI160 has a hardware-selectable I^2C slave address, depending on the logic driven on the SDO pin. On the E-INK Display Shield, the SDO pin is pulled to GND, which selects the slave address 0b1101000 (0x68).

BMI160 provides two output pins (INT1 and INT2) to which various interrupt events can be assigned. In this example, the Step Detector event is assigned to INT1 and Orientation interrupt is assigned to INT2. On the E-INK Display Shield, INT1 and INT2 pins are connected to pin 2 and 1 respectively of J3. On the Pioneer Baseboard, INT1 and INT2 connects to P13[1] and P13[0] of PSoC 6 MCU. See the BMI160 datasheet for more details on interrupt outputs.

The BMI160 step detector interrupt output has configurable active level and pulse width. In this example, the INT1 output is configured to provide a rising-edge signal with a pulse width of 2.5 ms. On PSoC 6 MCU, P13[1] and P13[2] are configured as an input pin and is internally pulled down. The interrupt is used to detect when new steps are detected the step counts are fetched by reading a register using I^2C. Raw accelerometer data is read and processed on orientation interrupt to compute the orientation. The orientation is represented using a set of graphic icons (arrows) on the E-INK display.

In PSoC Creator, an SCB-based I^2C Component is used to implement the I^2C Master interface to BMI160. The I^2C Data Rate is set to 400 kbps. Configuration of the motion sensor and reading accelerometer and step count information are performed over this interface.

Step counts are polled whenever the step detector interrupt is detected. A green LED (StepDetected_LED pin / P1[1]) is toggled with every step detected.

Figure 6. PSoC Creator Schematic Showing Motion Sensor Interface and Debug Outputs

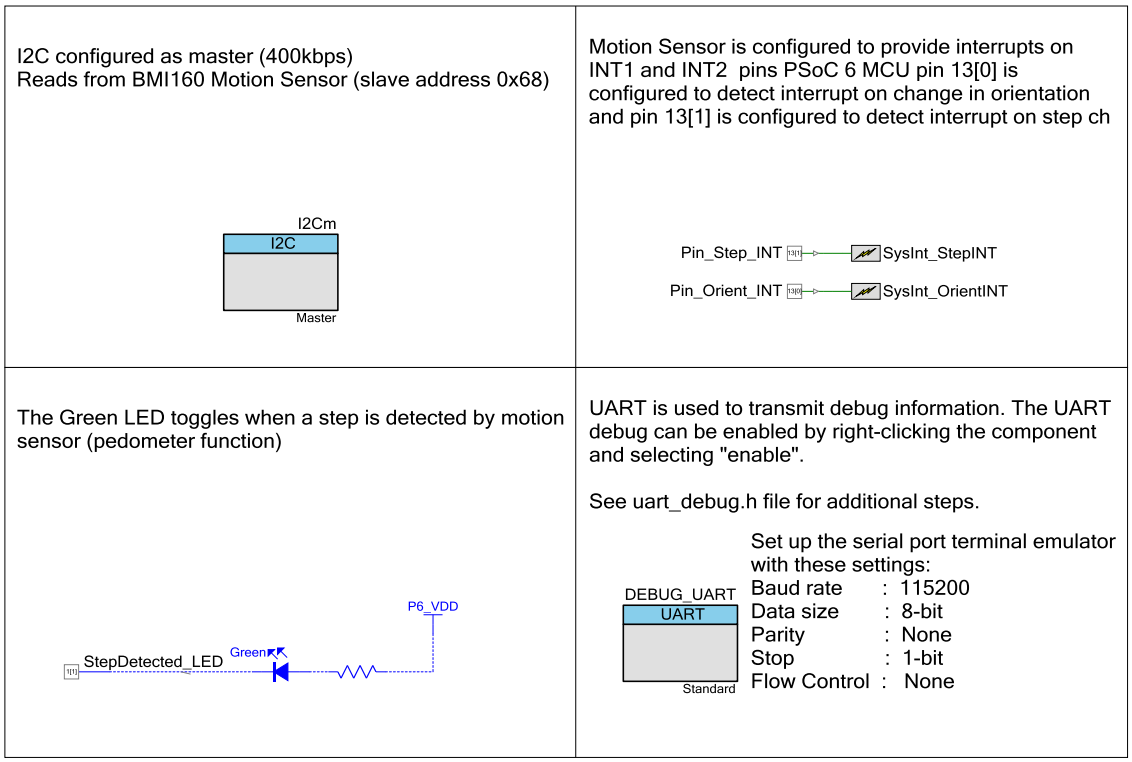


Figure 7. PSoC Creator Schematic: E-INK Display

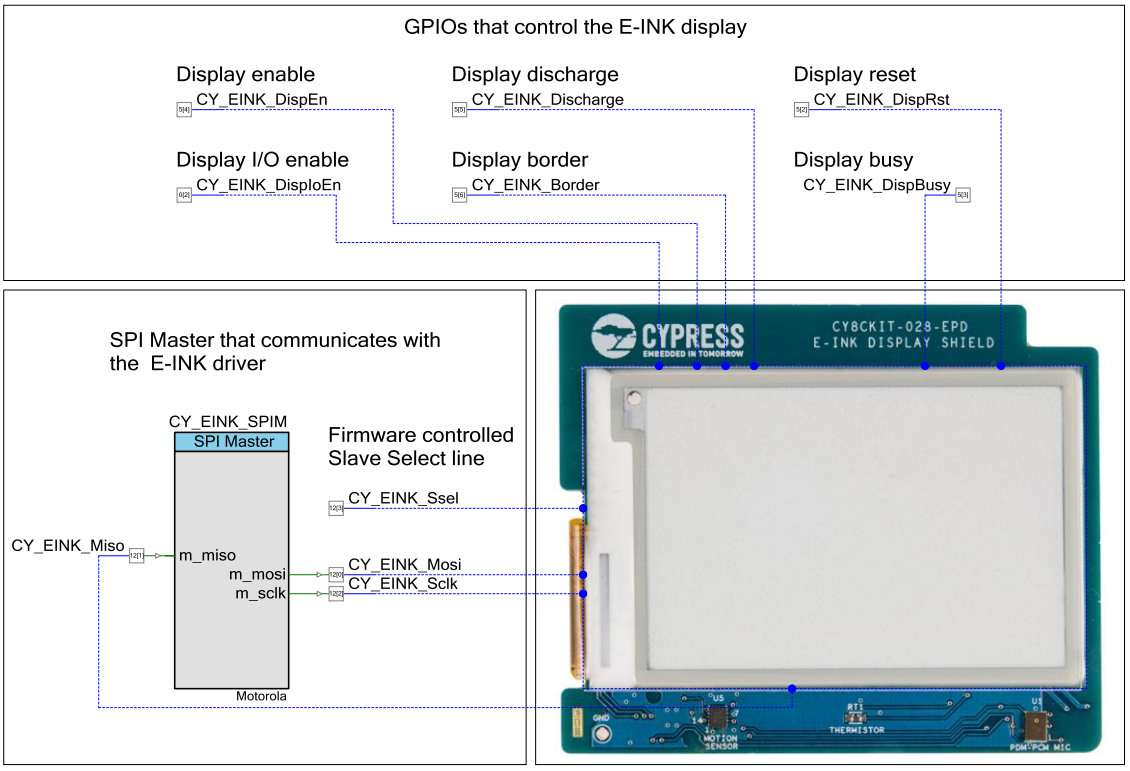
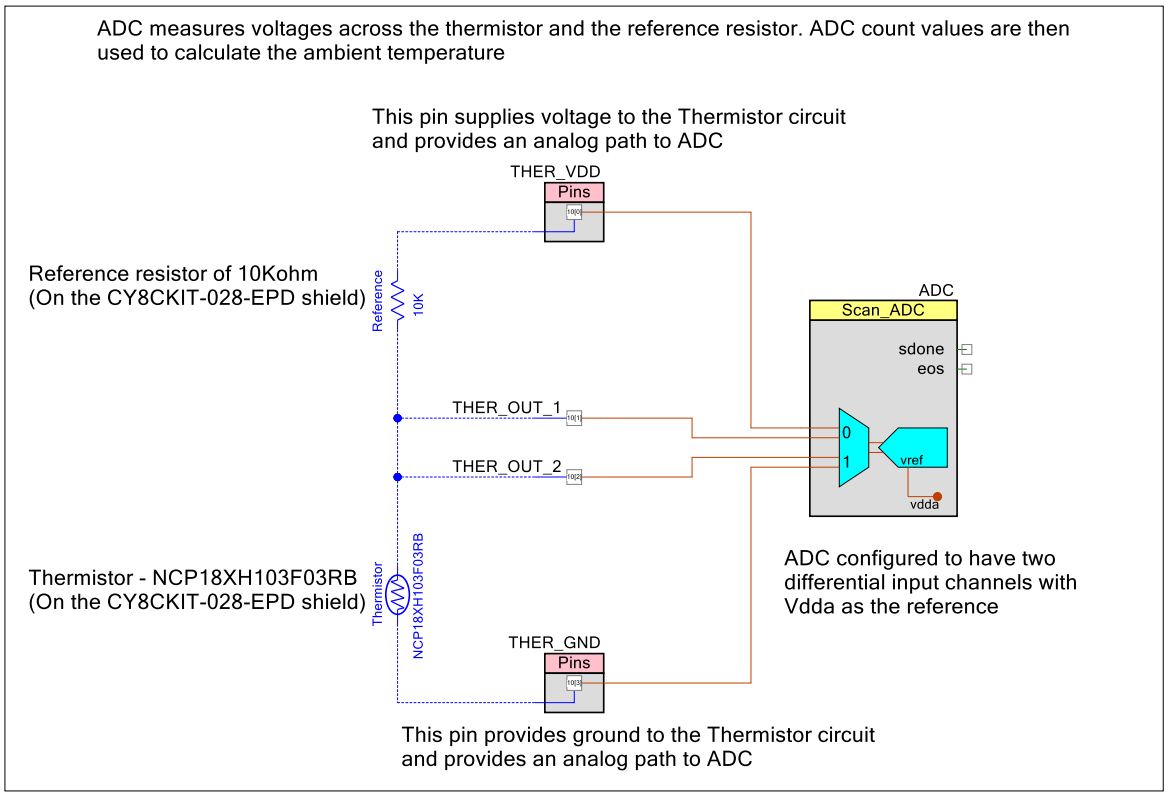


Figure 8. PSoC Creator Schematic: Temperature Compensation



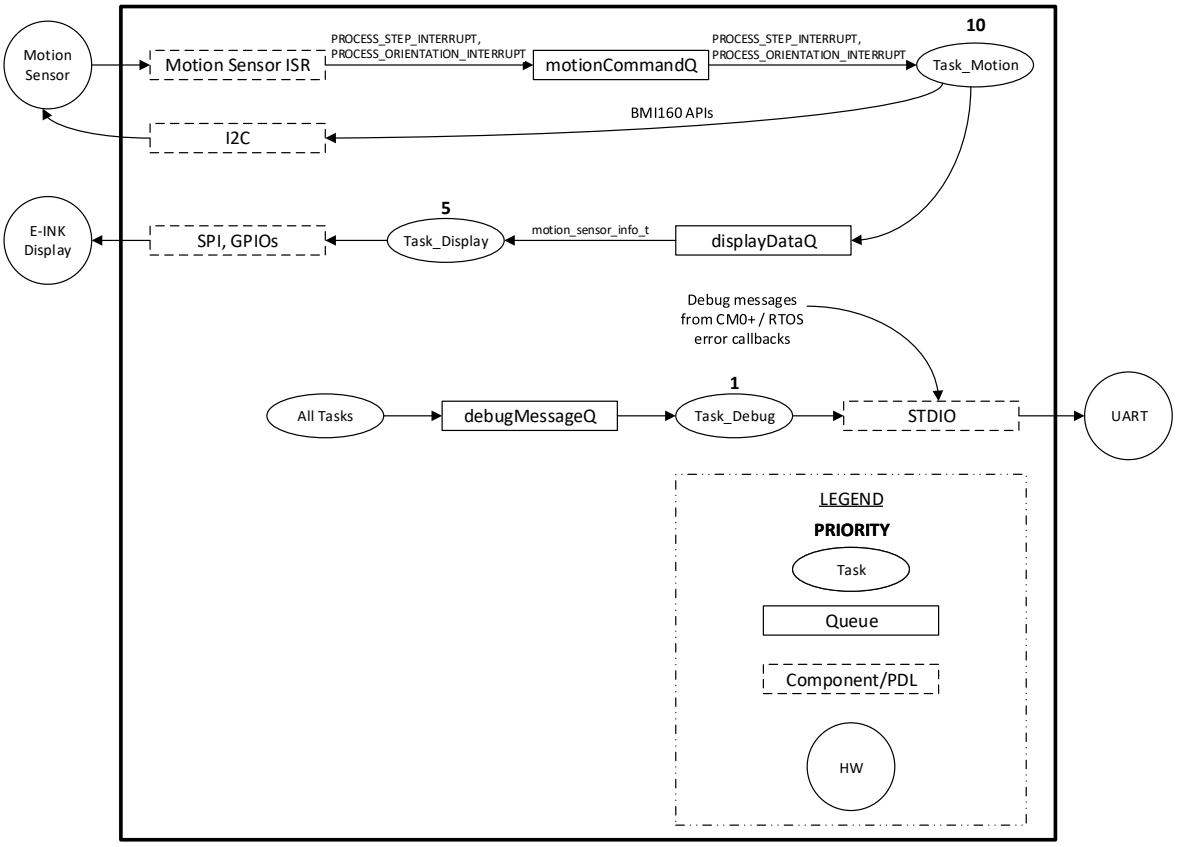
The project consists of the following files:

- *FreeRTOSConfig.h* contains the FreeRTOS settings and configuration. Non-default settings are explained with in-line comments.
- *main_cm4.c* contains the main function for Arm Cortex®-M4, which is the entry point and execution of the firmware application. The main function sets up user tasks and then starts the RTOS scheduler.
- *main_cm0p.c* enables CM4.
- *bmi160.c/.h*, *bmi160_def.h* contain the sensor driver for BMI160 motion sensor. For the latest version of this driver, visit the GitHub repository.
- *task_motion.c/.h* contain the task and macro definitions related to motion sensor application outputs. This includes the functions used to initialize and configure the motion sensor, set up interrupts, and compute motion outputs like orientation from raw accelerometer data.
- *i2cm_support.c/.h* contain the I^2C master read and write functions.
- *task_display.c/.h* contain the task that initialize the E-INK display and show the instructions to use this code example at startup[1].
- *temperature_eink.c/.h* contain functions that measure ambient temperature for E-INK display compensation
- *stdio_user.c/.h* contain functions for the debug and UART functionality.
- uart_debug.c/h contain the task and functions that enable UART based debug message printing.

Figure 9 shows the RTOS firmware flow for this project.

[1] For a detailed list of files included in the E-INK Library, see the code example CE218133 PSoC 6 MCU E-INK Display with CapSense.

Figure 9. Firmware Flowchart



Components and Settings

Table 3 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 3. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
I2C (SCB)	I2Cm	I^2C master for communicating with the motion sensor	Mode: Master Data Rate (kbps): 400
Interrupt	SysInt_StepINT	Component to receive signal from Pin_Step_INT	Interrupt Type: Rising-Edge Triggered
	SysInt_OrientINT	Component to receive signal from Pin_Orient_INT	
UART (SCB)	DEBUG_ UART	Serial communication block for debug output on terminal	Default
Digital Output Pin	StepDetected_LED	Visual output of step detection	Initial drive state: High (1)
Digital Input Pin	Pin_Step_INT Pin_Orient_INT	Pin connected to the motion sensor interrupt signal	[General tab] Drive mode: Resistive Pull Down [Input tab] Interrupt: Rising Edge

Note: See the code example CE220335 – PSoC 6 MCU E-INK Display with CapSense (RTOS) for more details on components used by E-INK library.

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

Reusing This Example

This example is designed for the PSoC 6 BLE Pioneer Kit. To port the design to a different PSoC 6 MCU device, kit, or both, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed. For single-core PSoC 6 MCU devices, port the code from *main_cm4.c* to *main.c*.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN215656 – PSoC 6 MCU: Dual-Core CPU system Design	Describes the dual-core CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
PSoC Creator Component Datasheets	
I2C	Supports I^2C slave, master, and master-slave operation configurations using SCB
SPI	Provides an industry-standard, 4-wire master SPI interface using SCB hardware
UART	Provides asynchronous communication interface using SCB hardware
Pins	Supports connection of hardware resources to physical pins
Interrupt	Provides Interrupt component settings
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
BMI160 Motion Sensor datasheet	PSoC 6 MCU: PSoC 62 Datasheet
Development Kit Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE222793 – PSoC 6 MCU: Motion Sensor (RTOS)

Document Number: 002-22793

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6101032	ARVI, AJYA	02/15/2018	New code example
*A	6142678	AJYA	04/17/2018	Updated project folder structure
*B	6302604	AJYA	09/10/2018	Added temperature compensation for E-Ink Display

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 | 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

© Cypress Semiconductor Corporation, 2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

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.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.