# Interprocess Communication

## Purpose

This lab will demonstrate some principles of interprocess communication using FreeRTOS.

## You Should Turn In

1. Answers to Q1, Q2, Q3.

As always, output snapshots are cool.

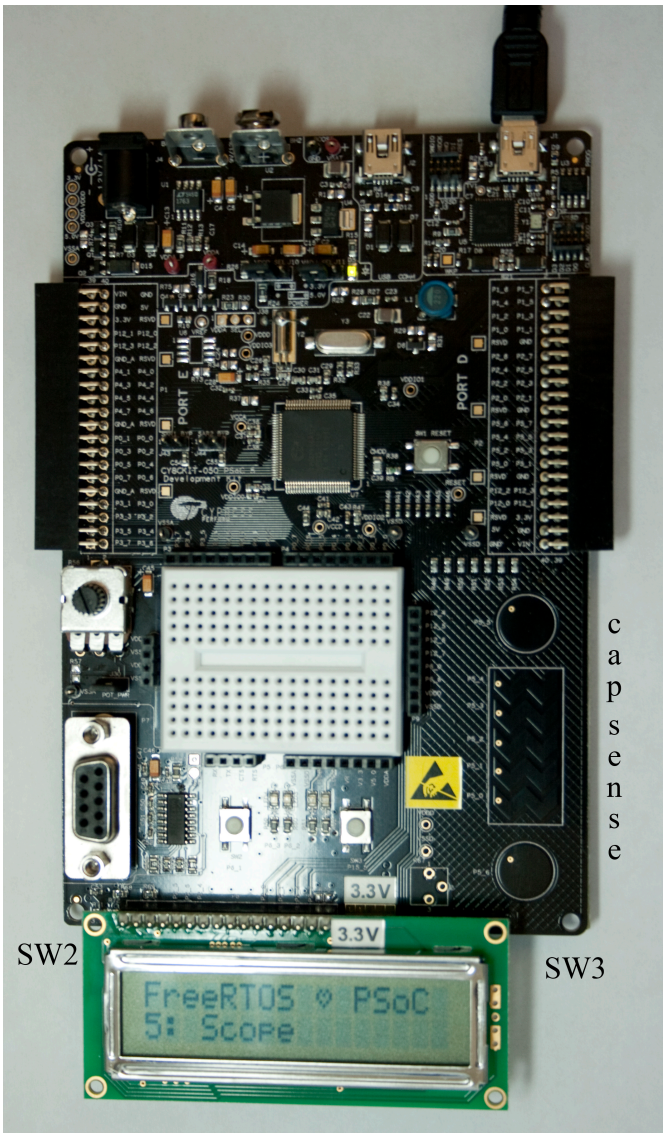
Extra credit: replace the shared variable for interprocess communication with the FreeRTOS queue.

## Setup

Copy the archive into your directory. Unzip it, open it, and build it.

## Procedure

This application implements the Scope demo of the FreeRTOS demo using a very different software structure: one task reads the button, a separate task updates the display. The two tasks communicate using a shared variable. (You can also use the capsense slider to change the periods of the tasks.) Here is the layout of the board:




The tasks that implement the scope function are in Reader_Writer.c. Reader_Task() reads SW2 and puts the result in next_char; Writer_Task() copies next_char into the output buffer and updates the display.

Run the program and observe its operation as you press SW2. Use the capsense slider to change the task periods.

Q1: Draw a UML state diagram for main loop of Reader_Task.

Q2: Draw a UML sequence diagram that shows the interactions between the tasks. The sequence diagram should include SW2, the display, next_char, Reader_Task(), and Writer_Task().

Each task includes a call to vTaskDelay() at the end of the main loop. This call puts the task into the blocked state until its next period. Comment out the call to vTaskDelay() for Reader_Task().

Q2: How does the behavior of the application change when the call to vTaskDelay() in Reader_Task() is commented out?