```c
#include "sl_component_catalog.h"
#include "sl_system_init.h"
#include "app.h"
#if defined(SL_CATALOG_POWER_MANAGER_PRESENT)
#include "sl_power_manager.h"
#endif
#if defined(SL_CATALOG_KERNEL_PRESENT)
#include "sl_system_kernel.h"
#else // SL_CATALOG_KERNEL_PRESENT
#include "sl_system_process_action.h"
#endif // SL_CATALOG_KERNEL_PRESENT

#include "em_device.h"
#include "em_chip.h"

#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"

/*******************************************************************************//**
 * Extern Includes
 ******************************************************************************/
extern void task_A(), task_B(), task_C(), task_D();

SemaphoreHandle_t semA = NULL;
SemaphoreHandle_t semB = NULL;
SemaphoreHandle_t semC = NULL;
SemaphoreHandle_t semD = NULL;

int tick_count = 0;

void TaskA(void *params)
{
  (void) params; // suppress warning
  for(;;)
  {
    if(xSemaphoreTake(semA,portMAX_DELAY))
    {
      task_A(); // perform actual task
    }
  }
}

void TaskB(void *params)
{
  (void) params; // suppress warning
  for(;;)
  {
    if(xSemaphoreTake(semB,portMAX_DELAY))
    {
      task_B(); // perform actual task
    }
  }
}

void TaskC(void *params)
{
```

```
 58   (void) params; // suppress warning
 59   for(;;)
 60   {
 61     if(xSemaphoreTake(semC,portMAX_DELAY))
 62     {
 63       task_C(); // perform actual task
 64     }
 65   }
 66 }
 67
 68 void TaskD(void *params)
 69 {
 70   (void) params; // suppress warning
 71   for(;;)
 72   {
 73     if(xSemaphoreTake(semD,portMAX_DELAY))
 74     {
 75       task_D(); // perform actual task
 76     }
 77   }
 78 }
 79
 80 void vApplicationTickHook(void)
 81 {
 82   //used to increase the tick count when an ISR is done
 83   if((tick_count % 30) == 0)
 84   {
 85       xSemaphoreGiveFromISR(semA, NULL);
 86   }
 87   if((tick_count % 40) == 0)
 88   {
 89       xSemaphoreGiveFromISR(semB, NULL);
 90   }
 91   if((tick_count % 60) == 0)
 92   {
 93       xSemaphoreGiveFromISR(semC, NULL);
 94   }
 95   if((tick_count % 109) == 0)
 96   {
 97       xSemaphoreGiveFromISR(semD, NULL);
 98   }
 99   tick_count++;
100 }
101
102 int main(void)
103 {
104   // Vendor function to work around bugs in some versions of the hardware
105     CHIP_Init();
106
107     // Create a semaphore
108     semA = xSemaphoreCreateBinary();
109     semB = xSemaphoreCreateBinary();
110     semC = xSemaphoreCreateBinary();
111     semD = xSemaphoreCreateBinary();
112
113     //use xTaskCreate(function name, "string name", configMINIMAL_STACK_SIZE, NULL, PRIORITY,
    NULL);
```

```
114
115     xTaskCreate(TaskA,
116                 "TaskA",
117                 configMINIMAL_STACK_SIZE,
118                 NULL,
119                 4,
120                 NULL);
121
122     xTaskCreate(TaskB,
123                 "TaskB",
124                 configMINIMAL_STACK_SIZE,
125                 NULL,
126                 3,
127                 NULL);
128
129     xTaskCreate(TaskC,
130                 "TaskC",
131                 configMINIMAL_STACK_SIZE,
132                 NULL,
133                 2,
134                 NULL);
135
136     xTaskCreate(TaskD,
137                 "TaskD",
138                 configMINIMAL_STACK_SIZE,
139                 NULL,
140                 1,
141                 NULL);
142
143     vTaskStartScheduler();
144     //vApplicationTickHook();
145
146   while (1) {
147
148   }
149 }
150
```