```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 "segmentlcd.h"
#include "semphr.h"

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

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

SemaphoreHandle_t semLCD = NULL; // semaphore for LCD display
SemaphoreHandle_t semCD = NULL; // semaphore for CD

int tick_count = 0;
int prog = 0;

char s[8]; // string for LCD display
int n; // number for LCD display

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
    while(!xSemaphoreTake(semLCD,portMAX_DELAY)){} // wait for LCD semaphore
    SegmentLCD_ARing(prog,0); // turn off previous segment
    prog = (prog +1) & 7;
    SegmentLCD_ARing(prog,1); // turn on next segment
    xSemaphoreGive(semLCD); // give LCD semaphore
    }
  }
}

void TaskC(void *params)
{
  (void) params; // suppress warning
  for(;;)
  {
    if(xSemaphoreTake(semC,portMAX_DELAY))
    {
      while(!xSemaphoreTake(semCD,portMAX_DELAY)){}
      task_C(s);
      xSemaphoreGive(semCD);
      while(!xSemaphoreTake(semLCD,portMAX_DELAY)){}
      SegmentLCD_Write(s);
      xSemaphoreGive(semLCD);
    }

  }
}

void TaskD(void *params)
{
  (void) params; // suppress warning
  for(;;)
  {
    if(xSemaphoreTake(semD,portMAX_DELAY))
    {
      while(!xSemaphoreTake(semCD,portMAX_DELAY)){}
      task_D(&n);
      xSemaphoreGive(semCD);
      while(!xSemaphoreTake(semLCD,portMAX_DELAY)){}
      SegmentLCD_Number(n);
      xSemaphoreGive(semLCD);
    }

  }
}

void vApplicationTickHook(void)
{
  //used to increase the tick count when an ISR is done
  if((tick_count % 3) == 0)
  {
      xSemaphoreGiveFromISR(semA, NULL);
  }
  if((tick_count % 125) == 0)
  {
      xSemaphoreGiveFromISR(semB, NULL);
  }
  if((tick_count % 29) == 0)
  {
      xSemaphoreGiveFromISR(semC, NULL);
  }
  if((tick_count % 49) == 0)
  {
      xSemaphoreGiveFromISR(semD, NULL);
  }
  tick_count++;
}

int main(void)
{
    // Vendor function to work around bugs in some versions of the hardware
    CHIP_Init();
    // Initialize the LCD
    SegmentLCD_Init(false);
    // Write to the display
    SegmentLCD_Number(0);
    SegmentLCD_Write("HELLO");

    // Create a semaphore
    semA = xSemaphoreCreateBinary();
    semB = xSemaphoreCreateBinary();
    semC = xSemaphoreCreateBinary();
    semD = xSemaphoreCreateBinary();

    semLCD = xSemaphoreCreateBinary();
    semCD = xSemaphoreCreateBinary();
    // Give the semaphore
    xSemaphoreGive(semLCD);
    xSemaphoreGive(semCD);

    //use xTaskCreate(function name, "string name", configMINIMAL_STACK_SIZE, NULL, PRIORITY, NULL);

    xTaskCreate(TaskA,
                "TaskA",
                configMINIMAL_STACK_SIZE,
                NULL,
                4,
                NULL);

    xTaskCreate(TaskB,
                "TaskB",
                configMINIMAL_STACK_SIZE,
                NULL,
                3,
                NULL);

    xTaskCreate(TaskC,
                "TaskC",
                configMINIMAL_STACK_SIZE,
                NULL,
                2,
                NULL);

    xTaskCreate(TaskD,
                "TaskD",
                configMINIMAL_STACK_SIZE,
                NULL,
                1,
                NULL);

    vTaskStartScheduler();
    //vApplicationTickHook();

  while (1) {}
}

```