

```

1#include "sl_component_catalog.h"
2#include "sl_system_init.h"
3#include "app.h"
4#if defined(SL_CATALOG_POWER_MANAGER_PRESENT)
5#include "sl_power_manager.h"
6#endif
7#if defined(SL_CATALOG_KERNEL_PRESENT)
8#include "sl_system_kernel.h"
9#else // SL_CATALOG_KERNEL_PRESENT
10#include "sl_system_process_action.h"
11#endif // SL_CATALOG_KERNEL_PRESENT
12
13#include "em_device.h"
14#include "em_chip.h"
15
16#include "FreeRTOS.h"
17#include "task.h"
18
19#include "segmentlcd.h"
20#include "semphr.h"
21
22/*****
23 * Extern Includes
24 * extern void task_A(void), task_B(void), task_C(char *), task_D(int *);
25 *****/
26extern void task_A(void), task_B(void), task_C(char*), task_D(int*);
27
28SemaphoreHandle_t semA = NULL;
29SemaphoreHandle_t semB = NULL;
30SemaphoreHandle_t semC = NULL;
31SemaphoreHandle_t semD = NULL;
32
33SemaphoreHandle_t semLCD = NULL; // semaphore for LCD display
34SemaphoreHandle_t semCD = NULL; // semaphore for CD
35
36int tick_count = 0;
37int prog = 0;
38
39char s[8]; // string for LCD display
40int n; // number for LCD display
41
42void TaskA(void *params)
43{
44    (void) params; // suppress warning
45    for(;;)
46    {
47        if(xSemaphoreTake(semA,portMAX_DELAY))
48        {
49            task_A(); // perform actual task
50        }
51    }
52}
53
54void TaskB(void *params)
55{
56    (void) params; // suppress warning
57    for(;;)

```

```
58 {
59     if(xSemaphoreTake(semB,portMAX_DELAY))
60     {
61         task_B(); // perform actual task
62         while(!xSemaphoreTake(semLCD,portMAX_DELAY)){ } // wait for LCD semaphore
63         SegmentLCD_ARing(prog,0); // turn off previous segment
64         prog = (prog +1) & 7;
65         SegmentLCD_ARing(prog,1); // turn on next segment
66         xSemaphoreGive(semLCD); // give LCD semaphore
67     }
68 }
69 }
70
71 void TaskC(void *params)
72 {
73     (void) params; // suppress warning
74     for(;;)
75     {
76         if(xSemaphoreTake(semC,portMAX_DELAY))
77         {
78             while(!xSemaphoreTake(semCD,portMAX_DELAY)){ }
79             task_C(s);
80             xSemaphoreGive(semCD);
81             while(!xSemaphoreTake(semLCD,portMAX_DELAY)){ }
82             SegmentLCD_Write(s);
83             xSemaphoreGive(semLCD);
84         }
85     }
86 }
87 }
88
89 void TaskD(void *params)
90 {
91     (void) params; // suppress warning
92     for(;;)
93     {
94         if(xSemaphoreTake(semD,portMAX_DELAY))
95         {
96             while(!xSemaphoreTake(semCD,portMAX_DELAY)){ }
97             task_D(&n);
98             xSemaphoreGive(semCD);
99             while(!xSemaphoreTake(semLCD,portMAX_DELAY)){ }
100             SegmentLCD_Number(n);
101             xSemaphoreGive(semLCD);
102         }
103     }
104 }
105 }
106
107 void vApplicationTickHook(void)
108 {
109     //used to increase the tick count when an ISR is done
110     if((tick_count % 3) == 0)
111     {
112         xSemaphoreGiveFromISR(semA, NULL);
113     }
114     if((tick_count % 125) == 0)
```

```
115 {
116     xSemaphoreGiveFromISR(semB, NULL);
117 }
118 if((tick_count % 29) == 0)
119 {
120     xSemaphoreGiveFromISR(semC, NULL);
121 }
122 if((tick_count % 49) == 0)
123 {
124     xSemaphoreGiveFromISR(semD, NULL);
125 }
126 tick_count++;
127 }
128
129 int main(void)
130 {
131     // Vendor function to work around bugs in some versions of the hardware
132     CHIP_Init();
133     // Initialize the LCD
134     SegmentLCD_Init(false);
135     // Write to the display
136     SegmentLCD_Number(0);
137     SegmentLCD_Write("HELLO");
138
139     // Create a semaphore
140     semA = xSemaphoreCreateBinary();
141     semB = xSemaphoreCreateBinary();
142     semC = xSemaphoreCreateBinary();
143     semD = xSemaphoreCreateBinary();
144
145     semLCD = xSemaphoreCreateBinary();
146     semCD = xSemaphoreCreateBinary();
147     // Give the semaphore
148     xSemaphoreGive(semLCD);
149     xSemaphoreGive(semCD);
150
151     //use xTaskCreate(function name, "string name", configMINIMAL_STACK_SIZE, NULL, PRIORITY,
    NULL);
152
153     xTaskCreate(TaskA,
154                 "TaskA",
155                 configMINIMAL_STACK_SIZE,
156                 NULL,
157                 4,
158                 NULL);
159
160     xTaskCreate(TaskB,
161                 "TaskB",
162                 configMINIMAL_STACK_SIZE,
163                 NULL,
164                 3,
165                 NULL);
166
167     xTaskCreate(TaskC,
168                 "TaskC",
169                 configMINIMAL_STACK_SIZE,
170                 NULL,
```

main.c

Tuesday, October 31, 2023, 4:09 PM

```
171         2,  
172         NULL);  
173  
174     xTaskCreate(TaskD,  
175                 "TaskD",  
176                 configMINIMAL_STACK_SIZE,  
177                 NULL,  
178                 1,  
179                 NULL);  
180  
181     vTaskStartScheduler();  
182     //vApplicationTickHook();  
183  
184     while (1) {}  
185 }  
186
```