

Ayden Dauenhauer and Alan Rieger

Prof. Wolfe

ELEN 122L Tuesday 2:15 p.m.

7 May 2024

Lab 5 – Concurrency and Buffer Management

Project 1:

Main Code:

```
/* Private user code -----*/
/* USER CODE BEGIN 0 */
int count=0;
int buffer[64];
int snapshot[64];
int head=0;
int tail=0;
int temp=0;
/* USER CODE END 0 */

83 int main(void)
84 {
85     /* USER CODE BEGIN 1 */
86
87     /* USER CODE END 1 */
88
89     /* MCU Configuration-----*/
90
91     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
92     HAL_Init();
93
94     /* USER CODE BEGIN Init */
95
96     /* USER CODE END Init */
97
98     /* Configure the system clock */
99     SystemClock_Config();
100
101    /* Configure the peripherals common clocks */
102    PeriphCommonClock_Config();
103
104    /* USER CODE BEGIN SysInit */
105
106    /* USER CODE END SysInit */
107
108    /* Initialize all configured peripherals */
109    MX_GPIO_Init();
110    MX_LCD_Init();
111    MX_SAI1_Init();
112    MX_RTC_Init();
113    MX_TIM3_Init();
114    MX_TIM6_Init();
115    /* USER CODE BEGIN 2 */
116    HAL_TIM_Base_Start_IT(&htim3);
117    HAL_TIM_Base_Start_IT(&htim6);
118    /* USER CODE END 2 */
119
120    /* Infinite loop */
121    /* USER CODE BEGIN WHILE */
122    while (1)
123    {
124        /* USER CODE END WHILE */
125
126        /* USER CODE BEGIN 3 */
127    }
128    /* USER CODE END 3 */
129}
130
```

Interrupt Code:

```
/* USER CODE BEGIN PV */
extern int count;
extern int buffer[64];
extern int snapshot[64];
extern int head;
extern int tail;
extern int temp;
int num=0;
/* USER CODE END PV */

211 void EXTI0_IRQHandler(void)
212 {
213     /* USER CODE BEGIN EXTI0_IRQn 0 */
214
215     /* USER CODE END EXTI0_IRQn 0 */
216     HAL_GPIO_EXTI_IRQHandler(Joystick_Center_Pin);
217     /* USER CODE BEGIN EXTI0_IRQn 1 */
218     temp=head;
219     int i=0;
220     while(temp!=tail){
221         snapshot[i]=buffer[temp];
222         temp=(temp+1)%64;
223         i++;
224     }
225     while(i<64){
226         snapshot[i]=0xfeedbeef;
227         i++;
228     }
229     /* USER CODE END EXTI0_IRQn 1 */
230 }

231 /**
232  * @brief This function handles TIM3 global interrupt.
233  */
234 void TIM3_IRQHandler(void)
235 {
236     /* USER CODE BEGIN TIM3_IRQn 0 */
237
238     /* USER CODE END TIM3_IRQn 0 */
239     HAL_TIM_IRQHandler(&htim3);
240     /* USER CODE BEGIN TIM3_IRQn 1 */
241     if((tail+1)%64!=head){
242         buffer[tail]=count;
243         tail=(tail+1)%64;
244         count++;
245         num++;
246     }
247     /* USER CODE END TIM3_IRQn 1 */
248 }
249

250 /**
251  * @brief This function handles TIM6 global interrupt, DAC channell and channel2 underrun error interrupts.
252  */
253 void TIM6_DAC_IRQHandler(void)
254 {
255     /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
256
257     /* USER CODE END TIM6_DAC_IRQn 0 */
258     HAL_TIM_IRQHandler(&htim6);
259     /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
260     if(num > 3) {
261         head=(head+4)%64;
262         num=num-4;
263     }
264     else if(num<4){
265         head=tail;
266         num=0;
267     }
268     /* USER CODE END TIM6_DAC_IRQn 1 */
269 }
270 }
```

Screenshot of Snapshot Memory:

The screenshot shows a memory dump titled "snapshot". The address bar at the top says "Address: snapshot". Below it is a large text area containing hex values. The values are mostly zeros, with some non-zero values appearing in groups. For example, there are several instances of the value "FEEDBEEF" appearing in sequence. The address column starts at 0x20000328 and ends at 0x20000628.

Address	Value
0x20000328	00000331 00000332 00000333 00000334 00000335 00000336 00000337 00000338 00000339 0000033A 0000033B 0000033C 0000033D 0000033E 0000033F 00000340
0x20000348	00000341 00000342 00000343 00000344 00000345 00000346 00000347 00000348 00000349 0000034A 0000034B 0000034C 0000034D 0000034E 0000034F 00000350
0x20000358	00000351 00000352 00000353 00000354 00000355 00000356 00000357 00000358 00000359 0000035A 0000035B 0000035C 0000035D 0000035E 0000035F 00000360
0x20000368	00000361 00000362 00000363 00000364 00000365 00000366 00000367 00000368 00000369 0000036A 0000036B 0000036C 0000036D FEEDBEEF FEEDBEEF FEEDBEEF
0x20000428	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x20000468	00000000 00000000 00000000 00000000 00000000 00000000 2000428 0800020D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200004A8	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200004E8	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x20000528	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x20000568	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200005A8	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x200005E8	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x20000628	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

In your lab report you must explain in detail how you know that no data structures are corrupted due to concurrency violations.

We know that the data structures are not corrupted because we know from the ARM Reference manual (p. 325-236) that EXTI0, the center button on the joystick, has a higher NVIC interrupt priority than the interrupts for timer 3 and timer 6. EXTI0 has a priority of 13 while Timer 3 has a priority of 36 and Timer 6 has a priority of 61 (lower number=higher priority). Therefore, each time you press the center joystick, the snapshot that you take interrupts both timers since it has a higher priority and thus the data in the snapshot is not corrupted.

Project 2:

Main Code:

```
/* Private user code ----- */
/* USER CODE BEGIN 0 */
int count=0;
int buffer[64];
int snapshot[64];
int head=0;
int tail=0;
int temp=0;
```

```
83 int main(void)
84 {
85     /* USER CODE BEGIN 1 */
86
87     /* USER CODE END 1 */
88
89     /* MCU Configuration-----*/
90
91     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
92     HAL_Init();
93
94     /* USER CODE BEGIN Init */
95
96     /* USER CODE END Init */
97
98     /* Configure the system clock */
99     SystemClock_Config();
100
101    /* Configure the peripherals common clocks */
102    PeriphCommonClock_Config();
103
104    /* USER CODE BEGIN SysInit */
105
106    /* USER CODE END SysInit */
107
108    /* Initialize all configured peripherals */
109    MX_GPIO_Init();
110    MX_LCD_Init();
111    MX_SAI1_Init();
112    MX_RTC_Init();
113    MX_TIM3_Init();
114    MX_TIM6_Init();
115    /* USER CODE BEGIN 2 */
116    HAL_TIM_Base_Start_IT(&htim3);
117    HAL_TIM_Base_Start_IT(&htim6);
118    /* USER CODE END 2 */
119
120    /* Infinite loop */
121    /* USER CODE BEGIN WHILE */
122    while (1)
123    {
124        __disable_irq();
125        temp=head;
126        int i=0;
127        while(temp!=tail){
128            snapshot[i]=buffer[temp];
129            temp=(temp+1)%64;
130            i++;
131        }
132        while(i<64){
133            snapshot[i]=0xfeedbeef;
134            i++;
135        }
136        __enable_irq();
137        HAL_Delay(1000);
138    }
139    /* USER CODE END WHILE */
140
141    /* USER CODE BEGIN 3 */
142
143    /* USER CODE END 3 */
144 }
```

Interrupts Code:

```
/* USER CODE BEGIN PV */
extern int count;
extern int buffer[64];
extern int snapshot[64];
extern int head;
extern int tail;
extern int temp;
int num=0;
/* USER CODE END PV */

/**
 * @brief This function handles TIM3 global interrupt.
 */
void TIM3_IRQHandler(void)
{
    /* USER CODE BEGIN TIM3_IRQHandler 0 */

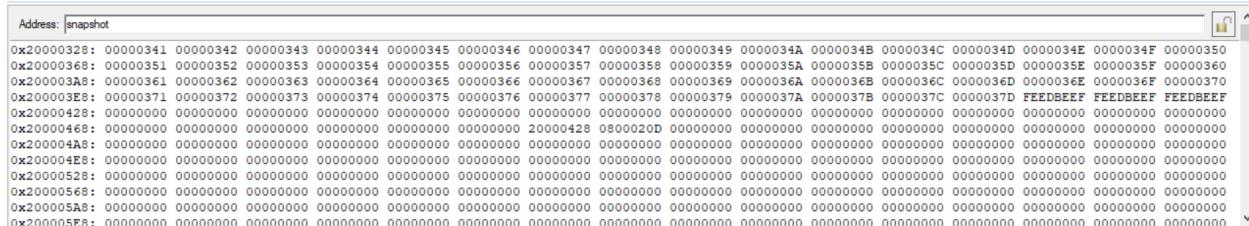
    /* USER CODE END TIM3_IRQHandler 0 */
    HAL_TIM_IRQHandler(&htim3);
    /* USER CODE BEGIN TIM3_IRQHandler 1 */
    if((tail+1)%64!=head){
        buffer[tail]=count;
        tail=(tail+1)%64;
        count++;
        num++;
    }
    /* USER CODE END TIM3_IRQHandler 1 */
}

/**
 * @brief This function handles TIM6 global interrupt, DAC channel1 and channel2 underrun error interrupts.
 */
void TIM6_DAC_IRQHandler(void)
{
    /* USER CODE BEGIN TIM6_DAC_IRQHandler 0 */

    /* USER CODE END TIM6_DAC_IRQHandler 0 */
    HAL_TIM_IRQHandler(&htim6);
    /* USER CODE BEGIN TIM6_DAC_IRQHandler 1 */
    if(num > 3) {
        head=(head+4)%64;
        num=num-4;
    }
    else if(num<4){
        head=tail;
        num=0;
    }
    /* USER CODE END TIM6_DAC_IRQHandler 1 */
}

/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

Screenshot:



In your lab report you must explain in detail how you know that no data structures are corrupted due to concurrency violations.

We know that the data structures were not corrupted because in the while loop in main, each time we take a snapshot of the buffer, we are disabling interrupts so that the timer interrupts (Timers 3 and 6) cannot interrupt us taking the snapshot. Thus, each time we take a snapshot (every second), we will never be interrupted until after we took the snapshot and enabled interrupts again.