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 */
```

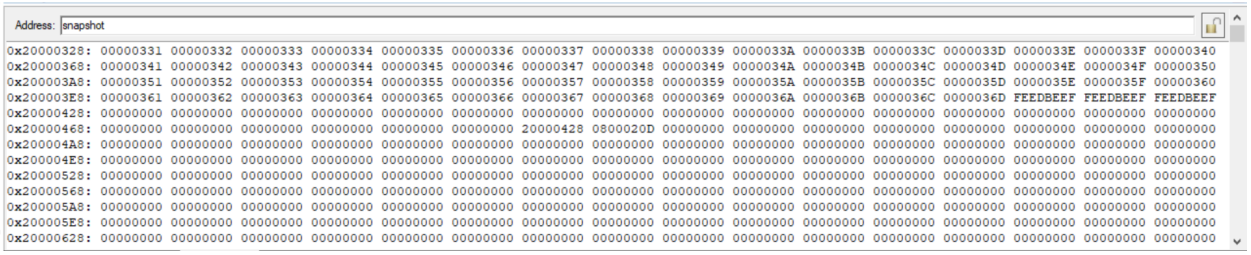
```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

/* Configure the peripherals common clocks */
  PeriphCommonClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_LCD_Init();
  MX_SAI1_Init();
  MX_RTC_Init();
  MX_TIM3_Init();
  MX_TIM6_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim3);
  HAL_TIM_Base_Start_IT(&htim6);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

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  /**
233    * @brief This function handles TIM3 global interrupt.
234    */
235  void TIM3_IRQHandler(void)
236  {
237    /* USER CODE BEGIN TIM3_IRQn 0 */
238
239    /* USER CODE END TIM3_IRQn 0 */
240    HAL_TIM_IRQHandler(&htim3);
241    /* USER CODE BEGIN TIM3_IRQn 1 */
242    if((tail+1)%64!=head){
243      buffer[tail]=count;
244      tail=(tail+1)%64;
245      count++;
246      num++;
247    }
248    /* USER CODE END TIM3_IRQn 1 */
249  }
250
251  /**
252    * @brief This function handles TIM6 global interrupt, DAC channel1 and channel2 underrun error interrupts.
253    */
254  void TIM6_DAC_IRQHandler(void)
255  {
256    /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
257
258    /* USER CODE END TIM6_DAC_IRQn 0 */
259    HAL_TIM_IRQHandler(&htim6);
260    /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
261    if(num > 3) {
262        head=(head+4)%64;
263        num=num-4;
264    }
265    else if(num<4){
266      head=tail;
267      num=0;
268    }
269    /* USER CODE END TIM6_DAC_IRQn 1 */
270  }
```

Screenshot of Snapshot Memory:



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:

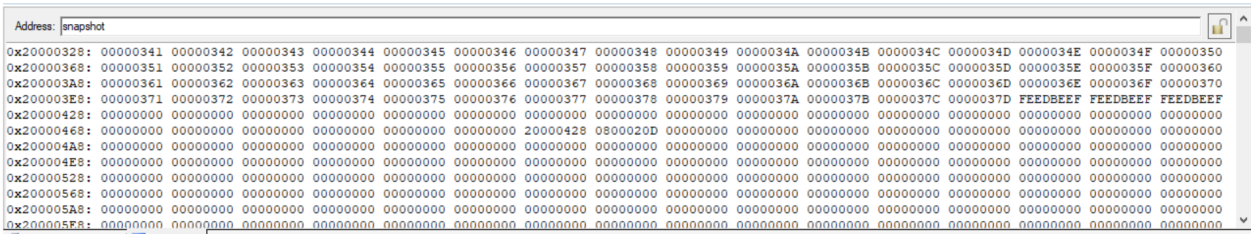
```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
int count=0;
int buffer[64];
int snapshot[64];
int head=0;
int tail=0;
int temp=0;
```

```c
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

/* Configure the peripherals common clocks */
  PeriphCommonClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_LCD_Init();
  MX_SAI1_Init();
  MX_RTC_Init();
  MX_TIM3_Init();
  MX_TIM6_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim3);
  HAL_TIM_Base_Start_IT(&htim6);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    __disable_irq();
    temp=head;
    int i=0;
    while(temp!=tail){
      snapshot[i]=buffer[temp];
      temp=(temp+1)%64;
      i++;
    }
    while(i<64){
      snapshot[i]=0xfeedbeef;
      i++;
    }
    __enable_irq();
        HAL_Delay(1000);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

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_IRQn 0 */

  /* USER CODE END TIM3_IRQn 0 */
  HAL_TIM_IRQHandler(&htim3);
  /* USER CODE BEGIN TIM3_IRQn 1 */
  if((tail+1)%64!=head){
    buffer[tail]=count;
    tail=(tail+1)%64;
    count++;
    num++;
  }
  /* USER CODE END TIM3_IRQn 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_IRQn 0 */

  /* USER CODE END TIM6_DAC_IRQn 0 */
  HAL_TIM_IRQHandler(&htim6);
  /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
  if(num > 3) {
      head=(head+4)%64;
      num=num-4;
  }
  else if(num<4){
    head=tail;
    num=0;
  }
  /* USER CODE END TIM6_DAC_IRQn 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.