MINI PROJECT 3

SPINLOCK SEMAPHORES & SLEEP FUNCTIONALITY

WILLIAM TRACE LACOUR (WTL6C)

PART 1 – QUESTION

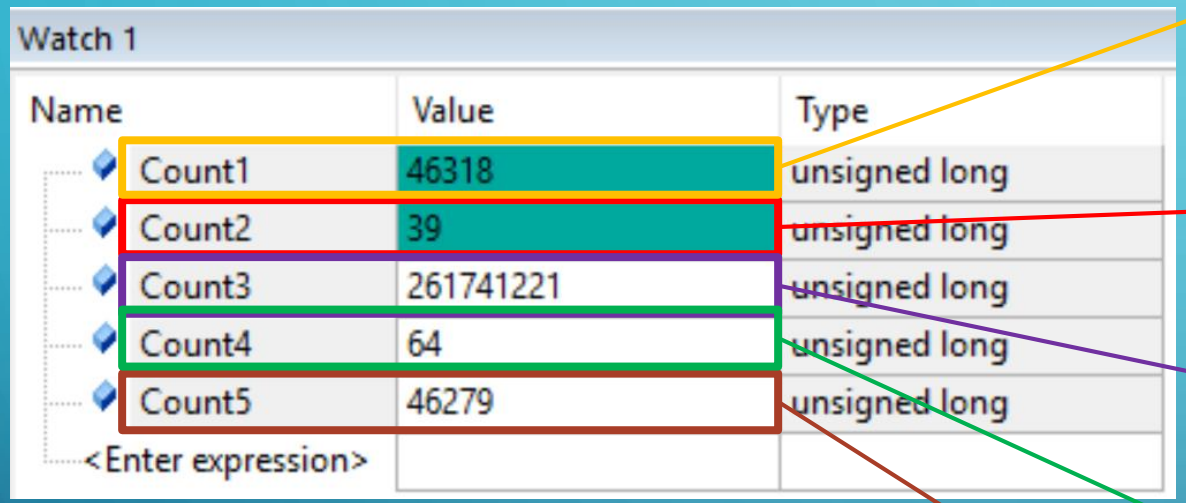
```
void Consumer(void){
  while(NumSamples < RUNLENGTH){
    jsDataType data;
    JsFifo_Get(&data);
    OS_bWait(&LCDFree);

    BSP_LCD_DrawCrosshair(prevx, prevy, LCD_BLACK); // Draw a black crosshair
    BSP_LCD_DrawCrosshair(data.x, data.y, LCD_RED); // Draw a red crosshair

    BSP_LCD_Message(1, 5, 3, "X: ", x);
    BSP_LCD_Message(1, 5, 12, "Y: ", y);
    OS_bSignal(&LCDFree);
    prevx = data.x;
    prevy = data.y;
  }
  OS_Kill();  // done
}
```

The Consumer function calls OS_bWait(&LCDFree) once entered. This is to check and make that the LCDFree binary semaphore is available for use and if it is not available, wait until it is. This ensures that when you're writing to the LCD screen that no else can have access to this section and do the same. Access to a section where writing or reading for that matter takes place, will result in faulty data. OS_bSignal(&LCDFree) functions allows for the semaphore to be signaled and the next thread waiting on this section can now have access. Overall this is a good practice for both protection and predictable data output. Using Wait and Signal allows for safe and reliable code. This is a excellent example of mutual exclusion and should be used during a critical section that should only have one thread running within it at a time.

PART 4 – TESTMAIN3



Count2 + Count5 = Count1
39 + 46279 = 46318

Count1 increments every time a signal is called. This will be the sum of both Count 5 and Count2.

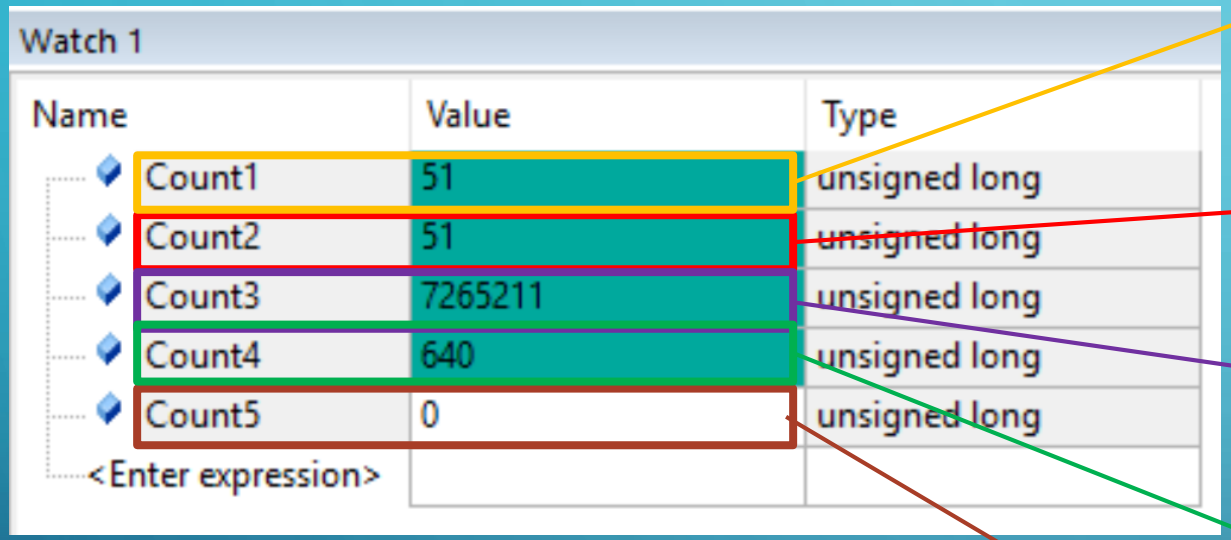
Count2 increments every time that (*semaPt).value = 0;

Count3 increments every time no other thread needs to run, this is why the value is so high.

Count4 increments by 64 every time that SW1 is pressed.

Count5 increments every time that (*semaPt).value > 0.

PART 4 – TESTMAIN4



Count1 = Count2
51 = 51

Count1 increments every 50ms and is acknowledged in Thread2. Count1 = Count2

Count2 increments every time it receives a signal from Thread1. Thread1 signals every 50ms.

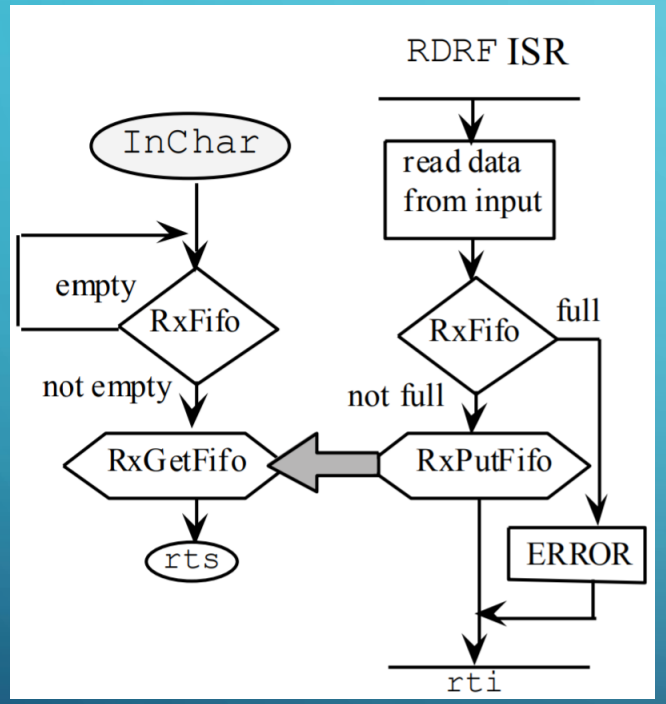
Count3 increments every time there is no other thread to run.

Count4 increments by 640 every time that the SW1 is pressed.

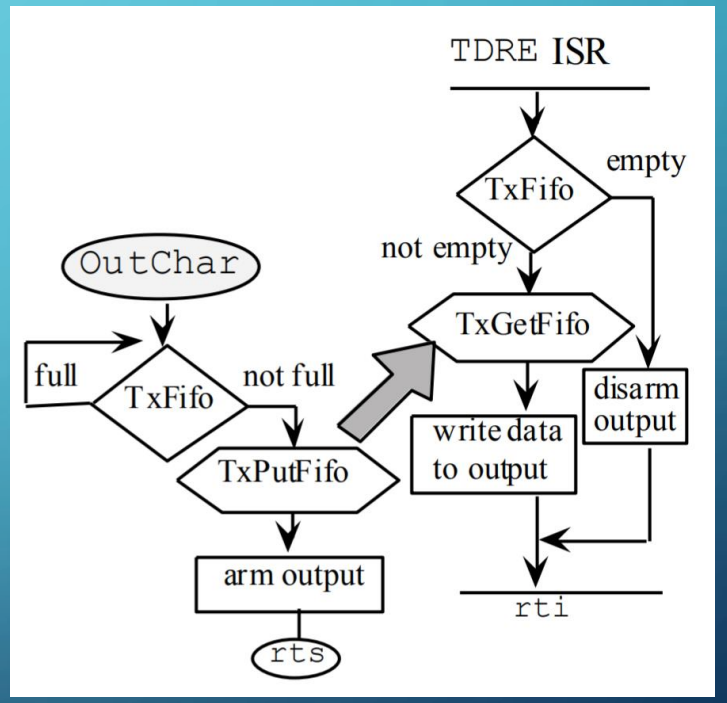
Count5 is not used in TESTMAIN4

PART 5 – FIFO SYNC FLOWCHART

Lecture 12 → Slide 21



Consumer Flowchart



Producer Flowchart

PART 5 – TABLE

FifoSize	TimeSlice	DataLost	Jitter	Calculations
8	2ms	140	8	35911954
32	2ms	0	3	34306531
64	2ms	0	3	34271876
64	1ms	0	6	33077368
64	10ms	0	1	34336876

I started with a fifo size of 8 but lost way to much data. I moved to a size of 32 and I didn't lose data at first but after repeated the test the data lost seemed random. At this point I moved to a fifo size of 64 and consistently lost 0 data. At this point I started to look at the Jitter and was different every time for the current 2ms time slice. I reduced the time slice to see if it got worse but it actually got better. I tried 10ms and was not as consistent as 1ms. The trend that I saw was that an increased fifo size decreased the amount of data lost and eventually approached zero data lost. The jitter on the other hand was very sporadic. I talked to other students and no one seemed to have that problem. I picked the values for fifo size and time slice based on the lowest value of jitter that I could get. The jitter did seem to be related to the amount of calculations that were made as well as the amount of threads created using the push button. The longer the time slice seemed to also make the jitter decrease as long as no data was lost because losing data also increased jitter (not seen in the table).

VIDEO

CHECK GITHUB FOR THE VIDEOLINK.TXT FILE FOR THE VIDEO

SURVEY SCREENSHOT