Exercise 15 Mutex

In this exercise our program writes streams of characters to the microcontroller UART from different threads. We will declare and use a mutex to guarantee that each thread has exclusive access to the UART until it has finished writing its block of characters.

In the Pack Installer select "Ex 15 Mutex" and copy it to your tutorial directory.

This project declares two threads which both write blocks of characters to the UART. Initially, the mutex is commented out.

```
void uart_Thread1 (void *argument) { uint32_t

i;

 for (;;) {

            //osMutexAcquire(uart_mutex, osWaitForever);

for( i=0;i<10;i++)    SendChar('1');

            SendChar('\n');

            SendChar('\r');

            //osMutexRelease(uart_mutex);

  }}
```

In each thread the code prints out the thread number. At the end of each block of characters it then prints the carriage return and new line characters.

Build the code and start the debugger.

Open the UART1 console window with View\Serial Windows\UART #1

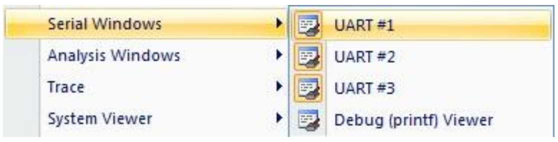
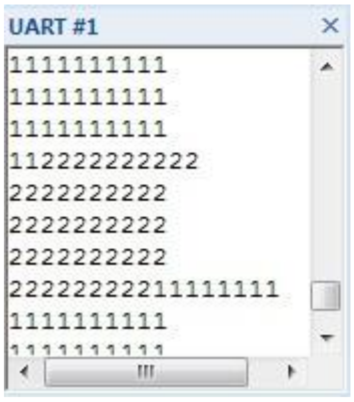


Fig 48 Open the UART console window

Start the code running and observe the output in the console window.



```
1111111111
1111111111
1111111111
112222222222
2222222222
2222222222
2222222222
22222222211111111
1111111111
1111111111
```

Fig 50 The mis-ordered serial output

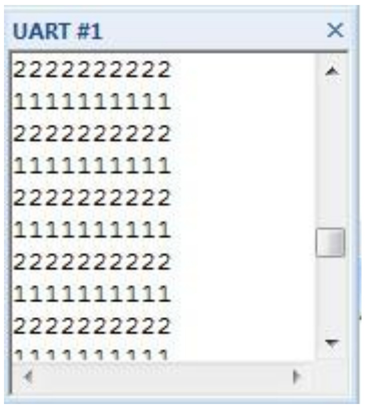
Here we can see that the output data stream is corrupted by each thread writing to the UART without any accessing control.

Exit the debugger.

Uncomment the mutex calls in each thread.

Build the code and start the debugger.

Observe the output of each task in the console window.



```
2222222222
1111111111
2222222222
1111111111
2222222222
1111111111
2222222222
1111111111
2222222222
1111111111
```

Fig 49 Order restored by using a mutex

Now the mutex guarantees each task exclusive access to the UART while it writes each block of characters.