Pomona College

Computer Science

Fall 2024: CSCI 181RT

Real-Time Systems in the Real World

Lecture 17

Thursday, October 24, 2024
Edmunds Hall 105
2:45 PM - 4:00 PM

Professor Jennifer DesCombes

Agenda

- Go Backs
- Discussion on Reading
- Lab #8 Review
- More Interrupts and OS Support
- Look Ahead
- Assignment
- Action Items

Go Backs

- General?
- Action Item Status
  - AI240910-2: Find recommended book on computer architecture.
  - AI240924-1: At what point as a development team grows does it make sense to have dedicated software and integration testers?

Discussion on Reading

- The Mythical Man Month
  - Chapter 19 & Epilog: *The Mythical Man Month* after 20 Years and the Epilogue

Lab #8 Review

- Optimization Must Be Removed
  - Looks Removed
  - Entire Subroutine Calls Removed
- Goals for Lab (from Lab 7 & 8)
  - Read Digital Input (GPIO1, Connector 501-Pin 5, Processor RK4)
  - Drive LED to Match Digital Input
- Sampling Rate and Data Input Rate (from Lab 7 & 8)
  - Use Function Generator to Experiment

Interrupts and OS Support

- OS Hardware Interrupts - Time and Timers ✓
- Peripheral Hardware Interrupts
  - Input Compare (IC) ✓
  - Serial Port (UART, USART)

Use of Semaphores - Shared Devices/Services - L12

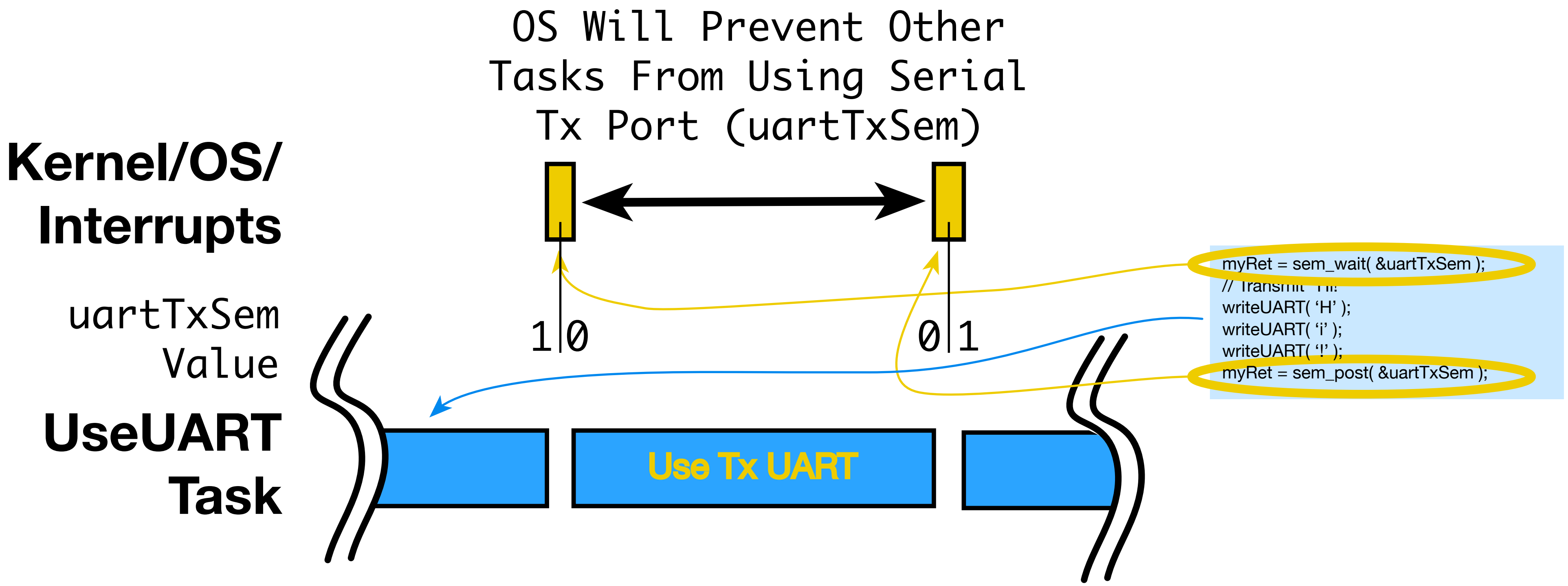
```
sem_t uartTxSem;
char uartChar;
void writeUART( char );

.
.
.
   int myRet;
   myRet = sem_wait( &uartTxSem );
   // Transmit “Hi!”
   writeUART( ‘H’ );
   writeUART( ‘i’ );
   writeUART( ‘!’ );
   myRet = sem_post( &uartTxSem );
.
.
.
```

If the UART is available, execution will continue.

If the UART is busy, execution will be suspended (waiting) until UART is available. Code will resume execution when UART is no longer being used by other code/task.

Let the system/tasks know that the UART is now available

Use of Semaphores - Shared Devices/Services - L12


OS Will Prevent Other Tasks From Using Serial Tx Port (uartTxSem)
Kernel/OS/ Interrupts
uartTxSem Value
UseUART Task
1
0
0
1
Use Tx UART

```
myRet = sem_wait( &uartTxSem );
// Transmit 'Hi!'
writeUART( 'H' );
writeUART( 'i' );
writeUART( '!' );
myRet = sem_post( &uartTxSem );
```

Use of Semaphores - Shared Devices/Services - L12

- Multiple Tasks, Both Using the Tx Serial Port (writeUART)

```
// High Priority Serial Task
void hpsTask (void)
{
   int myRet;
   while( true )
   {
      // Process every 5ms
      myOSSleepms(5);
      doImportantStuff();
      myRet = sem_wait( &uartTxSem )
      // Transmit “Hi!”
      writeUART( ‘H’ );
      writeUART( ‘i’ );
      writeUART( ‘!’ );
      myRet = sem_post( &uartTxSem );
   }
```

```
// Low Priority Serial Task
void lpsTask (void)
{
   int myRet;
   while( true )
   {
      doLessImportantStuff();
      myRet = sem_wait( &uartTxSem );
      // Transmit “OK”
      writeUART( ‘O’ );
      writeUART( ‘K’ )
      myRet = sem_post( &uartTxSem );
   }
```

Pomona College

Use of Semaphores - Shared Devices/Services - L12

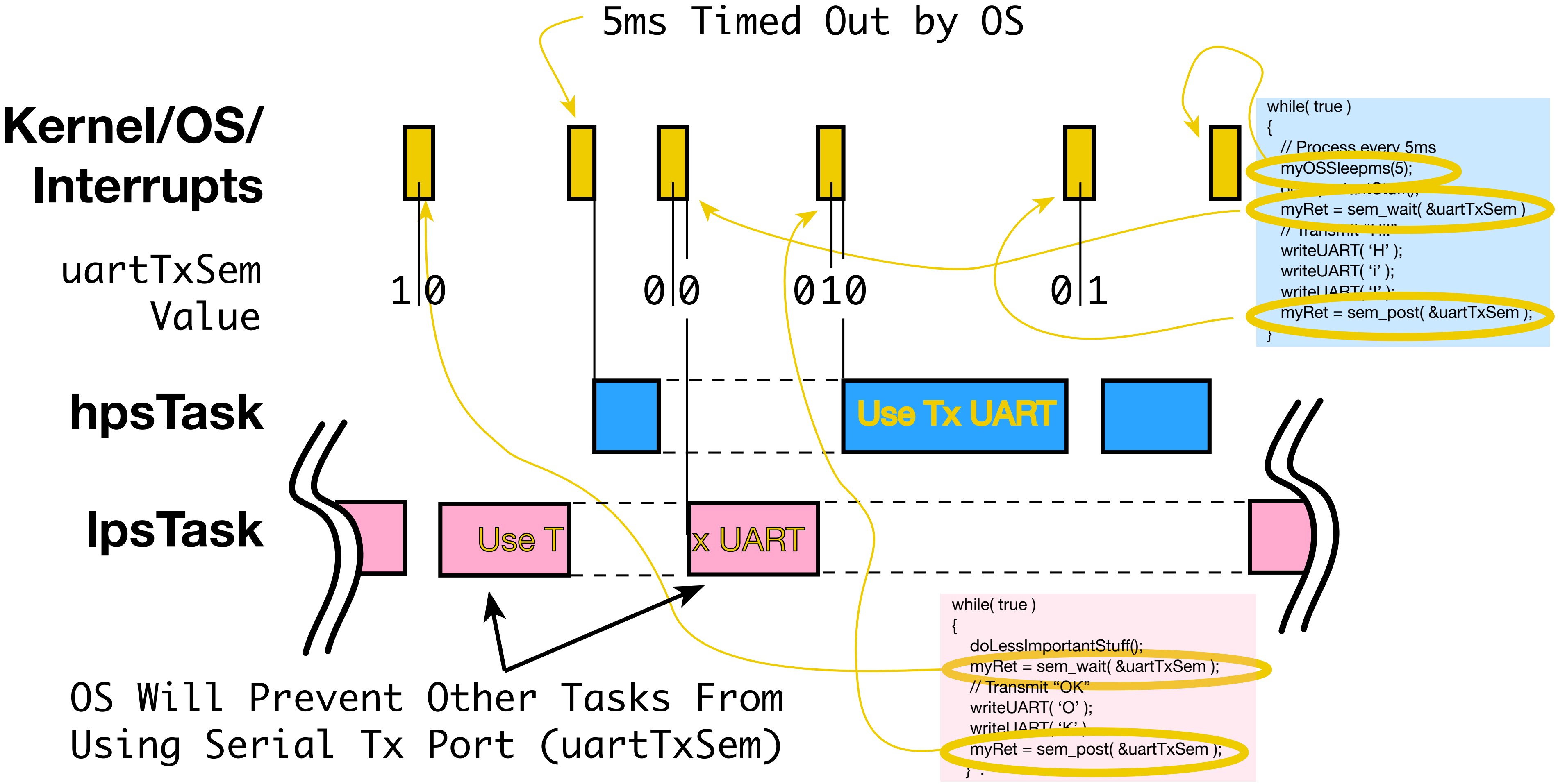
```
// High Priority Serial Task
void hpsTask (void)
{
   int myRet;
   while( true )
   {
      // Process every 5ms
      myOSSleepms(5);
      doImportantStuff();
      myRet = sem_wait( &uartTxSem )
      // Transmit “Hi!”
      writeUART( ‘H’ );
      writeUART( ‘i’ );
      writeUART( ‘!’ );
      myRet = sem_post( &uartTxSem );
   }
```

Semaphores control access. All calls use uartTxSem - OS prevents multiple access to UART (hardware device).

```
// Low Priority Serial Task
void lpsTask (void)
{
   int myRet;
   while( true )
   {
      doLessImportantStuff();
      myRet = sem_wait( &uartTxSem );
      // Transmit “OK”
      writeUART( ‘O’ );
      writeUART( ‘K’ )
      myRet = sem_post( &uartTxSem );
   }
```

Two Tasks (hpsTask and lpsTask) both use Tx Serial Port.

Use of Semaphores - Shared Devices/Services - L12



5ms Timed Out by OS

```
while( true )
{
  // Process every 5ms
  myOSSleepms(5);
  doImportantStuff();
  myRet = sem_wait( &uartTxSem )
  // Transmit “Hi!”
  writeUART( ‘H’ );
  writeUART( ‘i’ );
  writeUART( ‘!’ );
  myRet = sem_post( &uartTxSem );
}
```

```
while( true )
{
   doLessImportantStuff();
   myRet = sem_wait( &uartTxSem );
   // Transmit “OK”
   writeUART( ‘O’ );
   writeUART( ‘K’ );
   myRet = sem_post( &uartTxSem );
  } .
```

OS Will Prevent Other Tasks From
Using Serial Tx Port (uartTxSem)

Interrupts and OS Support - Serial Devices

- Types of Serial Interfaces
  - Transmit Function
    - UART Tx - RS-232, RS-422 (Point to Point)
  - Receive Function
    - UART Rx - RS-232, RS-422 (Point to Point)
  - Combined Functions - Single Logical Device
    - Point to Point - USB
    - Multidrop - RS-485, I2C, SPI, CAN,
- Synchronous Serial (U**S**ART) Can Be a Mixture of Both
  - Dependent on Generation and Use of Clock

Interrupts and OS Support - Serial Devices

- UART Transmit Status Signals and System Control
  - External Status
    - RTS - Ready to Send
    - CTS - Clear to Send
  - Internal Status
    - Currently Transmitting
    - Buffer Full (and how deep)
    - Buffer Empty
- Status Signals Can be Read from the Device
- Most Status Signals Can Generate Interrupts

Interrupts and OS Support - Serial Devices

- UART Receive Status Signals and System Control
  - External Status
    - RTS - Ready to Send
    - CTS - Clear to Send
  - Internal Status
    - Currently Receiving (start pulse detected)
    - Buffer Has Contents (and how deep)
    - Buffer Full and Overflow
- Status Signals Can be Read from the Device
- Most Status Signals Can Generate Interrupts

Interrupts and OS Support - Serial Devices

- Usage Models Will Drive System Architecture
- Does the Application Require Coordinated Tx and Rx
  - Commands with Acknowledgements
  - User Input and Output
  - These Usage Will Require Treating UART as Single Device
- Can Tx and Rx Be Independent
  - Tx to Informational Display
  - Rx Data Logger
  - These Usage Will Allow (not require) Treating UART as Dual Device

Interrupts and OS Support - Serial Devices

- Will be Using Single Device Architecture for Lecture/Discussion
- Simple Command and Acknowledgement Machine Control
- User Input and Output - Terminal (keyboard & monitor - CRT?)

Interrupts and OS Support - Serial Devices

- Simple Command and Acknowledgement Machine Control

```
sem_t uartTxNotBusy;
sem_t uartRxHasData;
char commandChar;
char responseChar;
char getCommand( void );
void processResponse( char );

.
.
   int myRet;
   myRet = sem_wait( &uartTxNotBusy );
   commandChar = getCommand( );
   writeUART( commandChar );
   myRet = sem_wait( &uartRxHasData );
   responseChar = readUART( );
   processResponse( responseChar );
.
.
```

Interrupt will occur when not transmitting and Tx buffer is empty

Interrupt will occur when a byte has been received.

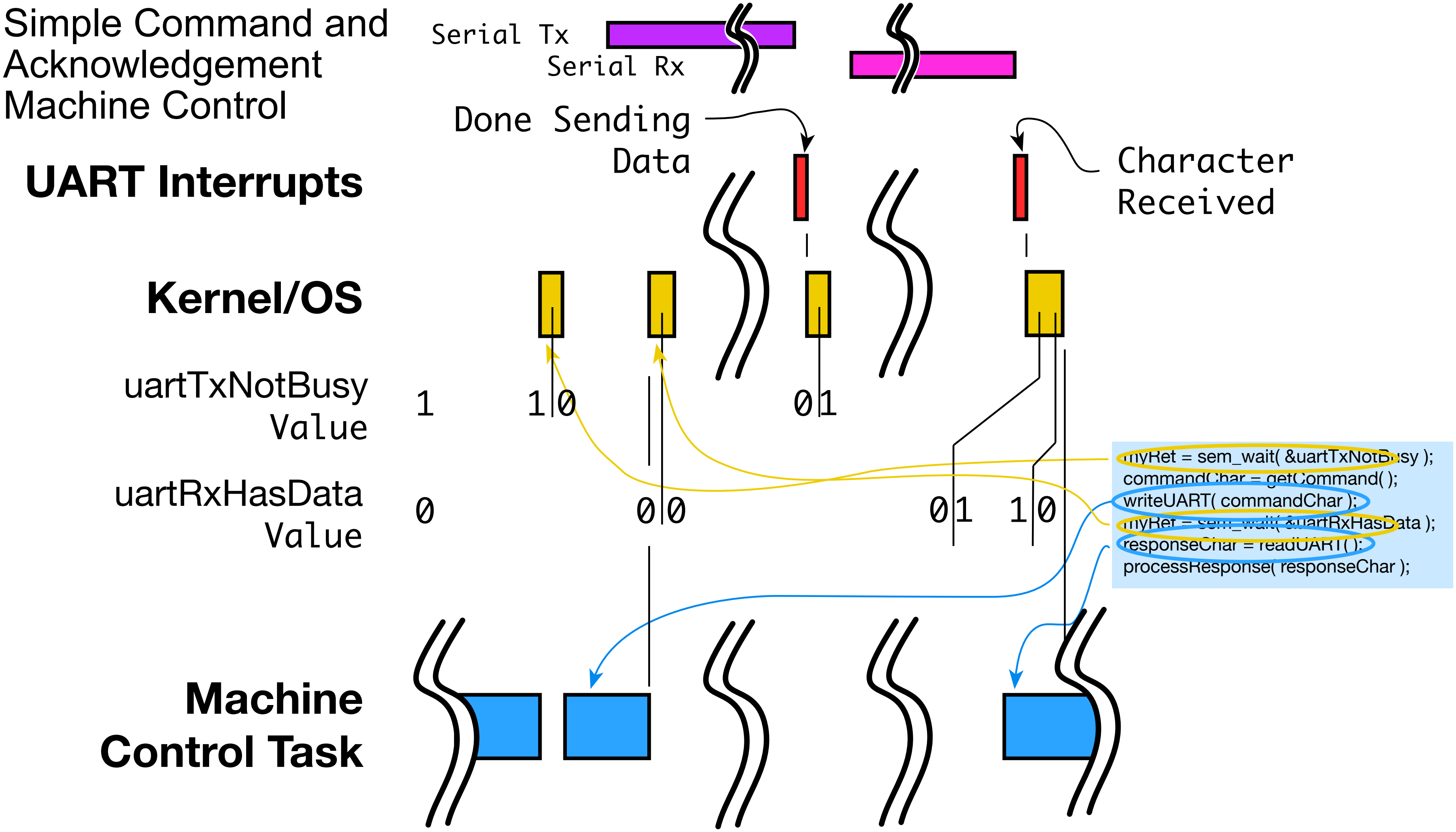
Wait, if necessary, for the Tx portion of the USART to complete all prior transactions.

Wait, if necessary, for the Rx portion of the USART to have received a character.

NOTE: Actual system design should include some sort of timeout on the acknowledgement .

Interrupts and OS Support - Serial Devices

- Simple Command and Acknowledgement Machine Control



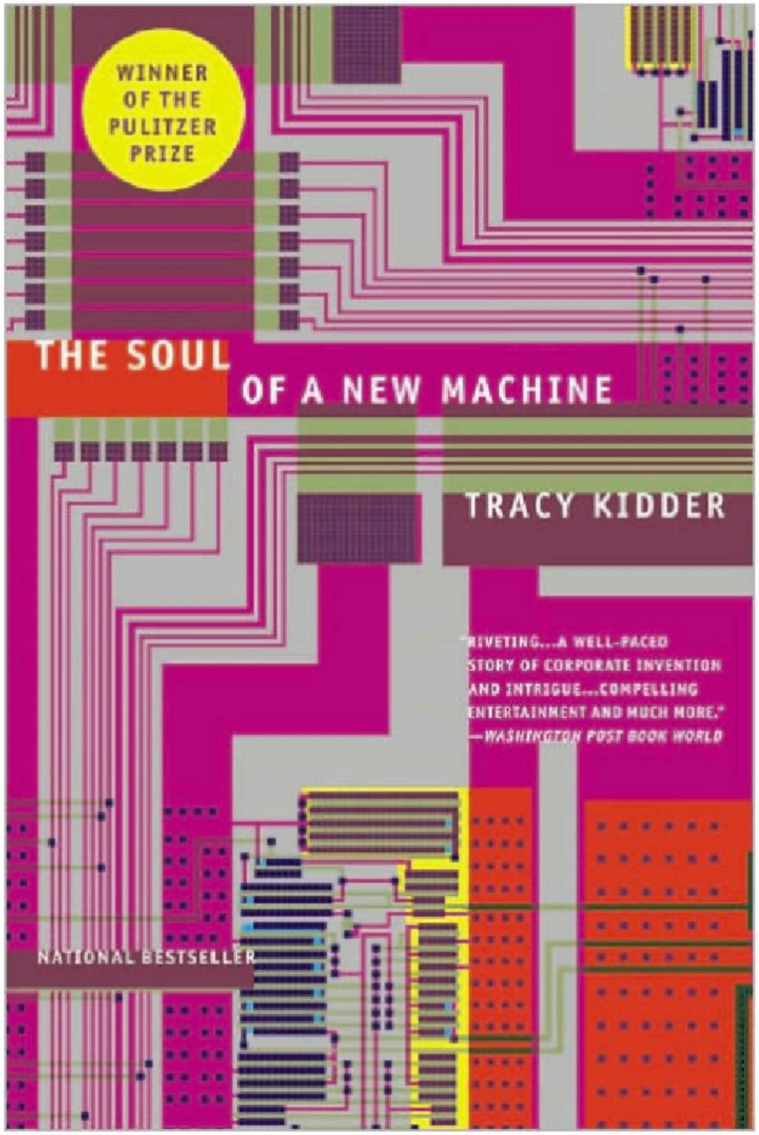
```
myRet = sem_wait( &uartTxNotBusy );
commandChar = getCommand( );
writeUART( commandChar );
myRet = sem_wait( &uartRxHasData );
responseChar = readUART( );
processResponse( responseChar );
```

Look Ahead

- Review of Reading
- Discussion of Code Framework
- Preview of Lab 9
- Discussion of Hardware Features and Capabilities - OS Related

Assignment - Readings

- The Soul Of A New Machine
  - Prologue, Chapter 1 and 2: How to Make a Lot of Money, The Wars.
  - Send Me Discussion Topics by 10:00 AM on Tuesday, Oct. 29, 2024.


WINNER OF THE PULITZER PRIZE
THE SOUL OF A NEW MACHINE
TRACY KIDDER
"RIVETING...A WELL-PACED STORY OF CORPORATE INVENTION AND INTRIGUE...COMPELLING ENTERTAINMENT AND MUCH MORE."
—WASHINGTON POST BOOK WORLD
NATIONAL BESTSELLER

Assignment - Code Review

- Application Description
- Template Application
  - What Didn’t You Like?
  - Why Did You Do That?
  - How Come?
- Have Comments Ready for Class
- Feedback Will be Incorporated into Template

Action Items and Discussion

AI#:	Owner	Slide #	Document	Action