Exercise in Embedded Computing: A UART Driver

Sivan Toledo, Tel-Aviv University

October 24, 2010

The Exercise

The purpose of this exercise is to develop a full UART driver `uart0.c` for the LPC2148. The driver uses two buffers, one for sending data and the other for receiving data. Here is the API that the driver should have.

void uart0Init() Initializes the UART.

int uart0TxBufferTrylock() Atomically tests for availability of the transmit buffer and acquires it if it is available. Return 1 if the buffer was acquired and 0 otherwise. To ensure atomicity, use `interruptsSaveAndDisable` to disable interrupts and `interruptsRestore` to restore them; both are defined in `interrupt.c`.

void uart0QueueByte(char c) Adds one character to the transmit buffer.

void uart0Send() Sends whatever has been queued in the transmit buffer. This function should asynchronous. That is, it should return almost immediately, not after all the data has been transmitted. While the data is being transmitted, the transmit buffer is unavailable.

int uart0RxBufferReady() Indicates whether a full frame is stored in the receive buffer. Frames end with `'\n'`.

char* uart0RxBuffer The address of the receive buffer. This can be a constant rather than a variable or a function.

int uart0RxBufferSize The size of the frame stored in the receive buffer, if it a full frame. If the buffer does not contain a full frame, this variable can store any value (client code should not use it when the buffer is not ready).

void uart0RxBufferRelease() This function signals to the driver that client code no longer needs the frame stored in the buffer. The buffer becomes not ready, and the driver can use it to store the next frame.

The driver should use interrupts to transmit and receive data. API functions should disable interrupts in order to ensure atomic access to the driver's global state variable. You should submit the driver and a test program, `test-uart-driver.c`. Start from the code in `exercises/uart`.