NOTES FOR CS452
UARTS ON THE EP9302

BILL COWAN
UNIVERSITY OF WATERLOO

A. INTRODUCTION

I did much of my fooling around with serial IO using an ipaq, which has a SA-1110 StrongARM processor and different UARTS than the ones on the EP9302. They are more different than I thought so that a couple of times I said things in class that are not quite correct. Below you will find clarifications that you might find useful.

B. TIMING

Internally the EP9302 UARTS are clocked by a clock that is 16 times the transmission speed: for example, 38.4 KHz for a UART communicating with the train at 2400 bits/sec. This, changes in the state of a UART can take quite a few CPU clocks before they occur. Thus, if you read the state of a UART too soon after writing a register that changes its state, you may not see the change in state.

The worst case occurs when transmitting to the train controller. In the worst case the state change happens 30 microseconds after the instruction that initiates it. Since the CPU, running with its caches turned on, can be executing 200 instructions per microseconds, the delay is quite a few instructions. Code that runs flawlessly with debugging output could fail intermittently when the debugging output is removed.

C. FLOW CONTROL

In class I suggested that the correct response to a modem status interrupt than de-asserts CTS is to turn off the transmitter so that so that characters in the FIFO will stop being sent. The SA-1110 has separate bits controlling the receiver and transmitter so that it is possible to disable the transmitter while the receiver continues to receive, but that EP9302 has only one bit that turns transmitting and receiving on and off together. Therefore, if you follow my suggestion and turn off the UART you might lose characters that are being sent to you by the train controller, without any evidence that input has been lost. This is not good. Here are two suggestions for working around this problem.

C.1. Disable the FIFOs.

You can disable the FIFOs on the UART that interacts with the train controller. Regrettably, it is not possible to disable on the transmitting FIFO as you can on the SA-1110, so you must do both receiving and transmitting one character at a time. This should not be too much of a problem because IO with the train controller happens so slowly.

In this work around you check CTS in the modem status register before transmitting as you did in Assignment 1, and defer transmitting until it is asserted again. Probably the best way to find out when is by enabling the modem status interrupt so that you don't waste time polling and can use the CPU for something else.

In fact, there is a variety of ways of using the modem status interrupt to make this work around more efficient. I suggest thinking about the transmit status in terms of a simple state diagram that you consult to determine whether or not it is safe to transmit.

C.2. Manipulate RTS.

If you want to use the FIFOs you can disable the UART and at the same time inhibit the train controller from sending to you by de-asserting RTS. At most one character could be on its way into the receiver, and you can pick it up later.

C.3. Summary.

I am certain that there exist other work arounds for this problem, which may work more reliably than the ones I have mentioned here. Finding them is a good skill to have for anyone who is doing low level programming.

As with many aspects of the programming in this course you need to experiment a little to find out exactly how the hardware behaves. In fact, part of the object of this course is to encourage you to supplement reading documentation, which is invariably incomplete, with experimentation to find out if your understanding of the documentation corresponds with reality.*

D. State Diagrams

An easy, and effective way of understanding your software is to create simple state diagrams. Components like Notifiers and Servers almost always implement transitions among a few states, triggered by messages or by changes in the hardware. Figure out how you want the states of the tasks to interact, and it will tell you how you should design the communication relationships.

* 'Reality' is always how the system behaves in practice, not how it should behave.