

# CSCE 491 Computer Engr. Design Project

2005/96

## Week 5 Modeling & Design Applications-2 The UART

© 2002 Dr. James P. Davis

---

---

---

---

---

---

---

---

---

---

### UART Function Definition-1

- Purpose.
  - ✓ Handle half-duplex serial data traffic between two connection points of a communications channel. Most UARTs are full-duplex, handling both send and received traffic simultaneously. We limit the scope to minimize complexity, to create a design model on which we can layer additional capabilities later.
  - ✓ This gives a very basic model of transmit/receive protocol handshaking, that will be the basis for examining the more complex handshaking associated with the 802.11 protocol.
- Algorithmic approach.
  - ✓ Use a pipelined "delegation" scheme, where the CPU enables the UART controller, and the controller enables either transmitter or receiver.
  - ✓ Data transfer between sender and receiver is done according to a 4-stage handshaking protocol:
    - ✓ RTS: sender endpoint requests to send data
    - ✓ CTS: receiver indicates that sender is "clear" to send the data stream.
    - ✓ The actual data is sent as a bit serial data stream, and the data carries additional parity bits.
    - ✓ ACK: receiver acknowledges that the data was received successfully.
  - ✓ We adopt a sequence of polling loops to synchronize activities of the protocol between sender and receiver end points.
- ASM diagram.
  - ✓ We create a design with 3 threads, plus a 4th thread that provides an abstraction of the CPU interface (with limited functionality to initiate activity in the UART blocks)
  - ✓ We'll use this 4th thread as a test harness.

---

---

---

---

---

---

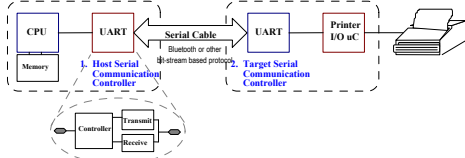
---

---

---

---

### UART Function Definition-2



#### Host Actions:

1. Set master enable; set control flags; load 32-bit data word for transmit sequence (LSB is parity bit).
3. Load 'transmit data' register; buffer with start, stop and parity bits; set 'ready to send' flag.
5. Reset counter; shift each of 10-bits onto serial bus; increment transfer counter.
8. Check and clear status flags; setup for next transmit.

#### Remote Actions:

2. Clear flags on Reset; drop 'clear to send' flag; poll for 'ready to send'.
4. Read 'ready to send' and acknowledge with 'clear to send' flag; set up the 'receive data counter'.
6. Shift in each of 10-bits from serial bus into buffer; check parity; strip off start and stop bits.
7. Check and clear status flags; transfer data to buffer.

---

---

---

---

---

---

---

---

---

---

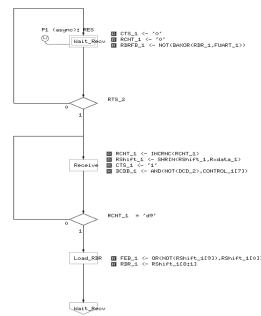








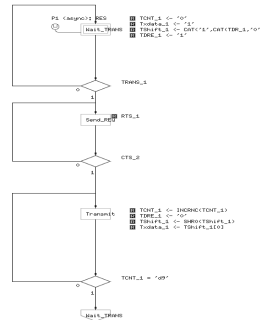
## UART Architecture - Receiver



- **Purpose.**
  - We model the basic Receiver functions, most of which deal with waiting for a new data stream and reading this stream, shifting the data, then signaling the Controller.
- **UART Receiver.**
  - We start up in a poll loop, waiting for the RTS signal, indicating we have a request from the remote host to send data to us.
  - The RTS\_2 signal controls whether we initiate preparation to receive data. One thing we do is set the CTS\_1 flag, initialize the RCNT counter, and shift in the serial bit.
  - We'll loop on the reading and shifting until the RCNT flag indicates we have a full data word (1 start bit, 7 ASCII bits, and 1 parity bit).
  - We then load the Read Data register RDR, and check the start and parity bits.

© 2002 Dr. James P. Davis Page 16

## UART Architecture - Transmitter



- **Purpose.**
  - We model the basic Transmitter functions to send data to the remote unit across the serial line.
- **UART Transmitter.**
  - The Transmitter waits to be signaled by the Controller, by the TRANS\_1 signal. A number of signals are sampled in Wait\_TRANS state continuously.
  - When enabled, the RTS for request to send signal is set, and we go into a poll loop awaiting the remote side to pull the CTS\_2 line, where we will let go of the RTS\_1 line.
  - We go into the Transmit poll loop, cycling this state to send out data, bit by bit, incrementing the bit counter on each cycle.
  - We complete when the counter reaches word length.

© 2002 Dr. James P. Davis Page 17

## UART - Design Verification

**CPU:** Simulation begins with CPU setting various control bits in the UART control register, illustrating the method in which data is transferred to the UART from the world and the ability for one component to control another through control signals.

**Controller:** After the master user places the data input on ACC bus and engages the CPU, the CPU enables the UART and sets the control lines, RS and RW, thus providing the UART instructions for properly dealing with the input.

**Transmit:** Once the control register is initialized, a transmit operation is executed. Again, the master user enables the CPU and provides data on the ACC bus. The UART Controller is engaged and the RS and RW lines are set to invoke a transmit operation. The Controller transfers the data from the UART bus to the Transmit Data Register, TDR, and the pulses TRANS, which enables the Transmitter. The Transmitter places the contents of the TDR to its shift register and begins iterative shifts on TShift, placing the LSB of the shift register on the TXdata line.

**Receive:** The master user sets the RTS\_2 input high indicating that remote device wishes to send data, and the Receiver is enabled. The Rxdata line is sampled ten times, one start bit, eight data bits, and one stop bit, and each bit is placed in its shift register. After the tenth sample, the RDRFB is set high to indicate the RDR has new data, and the data is transferred to the RDR after stripping off the start and stop bits. The CPU then engages a read cycle to retrieve the new data from the RDR, thus clearing the RDRFB signal.

**Parallel Transmit and Receive:** To prove that each of the components of the system may act in parallel, a simultaneous transmit and receive situation is initiated by the master user. The CPU is given the proper instructions to engage the UART Transmitter, at the same time, the RTS2 input goes high indicating a request to send data. A read cycle is performed after the receive operation to retrieve the new data from the RDR.



© 2002 Dr. James P. Davis Page 18

## UART – Summary

- The UART is a familiar component we have seen before in CPU-based peripheral interfacing applications.
  - ✓ CSCE 212 (Computer Architecture), CSCE 313 (Embedded Systems)
- We start with a basic abstract model, and develop a set of analysis “artifacts” for this component, including a test harness we’ll use to drive the design through a set of test scenarios.
  - ✓ The scenarios will come from analysis we do using Use Case and Sequence Diagrams.
  - ✓ We’ll build support for these test cases into an ASM thread for the CPU (a very abstract model of the CPU function as seen from its interface).
- Over the next few assignments, we’ll develop our understanding of this model, by refining and elaborating with more detailed functionality.
  - ✓ This serial bitstream communication model forms the basic pattern for our architecture analysis of the 802.11b protocol engine.



---

---

---

---

---

---

---

---

---

---