UART
Discussion III (Version 2.0)

UMBC - CE

September 16, 2014

Version 1.0 - Initial Document
Version 2.0 - Fixed typos and some illogical assumptions!
Objectives

- Introduce UART (Universal Asynchronous Receiver Transmitter)
Objectives

▶ Introduce UART (Universal Asynchronous Receiver Transmitter)

▶ Interface PC with AVR Butterfly via UART
Objectives

▶ Introduce UART (Universal Asynchronous Receiver Transmitter)
▶ Interface PC with AVR Butterfly via UART
▶ Implement UART communications using AVR Assembly
UART

- The simplest form of communication with a computer COM or serial port uses only 3 wires, GND, RX and TX (named with respect to the master)
The simplest form of communication with a computer **COM** or serial port uses only 3 wires, **GND**, **RX** and **TX** (named with respect to the master).

- Slave Device (TX) — Master Device (RX)
UART

- The simplest form of communication with a computer **COM** or serial port uses only 3 wires, **GND**, **RX** and **TX** (named with respect to the master)
- Slave Device (TX) → Master Device (RX)
- Master Device (TX) → Slave Device (RX)
UART

- The simplest form of communication with a computer **COM** or serial port uses only 3 wires, **GND**, **RX** and **TX** (named with respect to the master)
- Slave Device (TX) → Master Device (RX)
- Master Device (TX) → Slave Device (RX)
- The data is transmitted without a clock and is instead transmitted at a rate predetermined or pre-negotiated rate known on both sides
The simplest form of communication with a computer COM or serial port uses only 3 wires, GND, RX and TX (named with respect to the master).

Master Device (TX) → Slave Device (RX)

Slave Device (TX) → Master Device (RX)

The baud rate defines the length of each bit as
\[
\frac{1}{\text{baudrate}}
\]

The data is transmitted without a clock and is instead transmitted at a rate predetermined or pre-negotiated rate known on both sides.
The UART Frame

- The data to be sent is **framed** by a start bit and a stop bit.
The UART Frame

- The data to be sent is **framed** by a start bit and a stop bit

![Diagram showing UART frame with start bit, data bits, and stop bit]

- The start bit is typically a high signal to start the data frame, and the stop bit is typically a low signal, often 1, 1.5 or 2 times as long as the other bits.
The UART Frame

- The data to be sent is **framed** by a start bit and a stop bit

- The start bit is typically a high signal to start the data frame, and the stop bit is typically a low signal, often 1, 1.5 or 2 times as long as the other bits

- Optionally, a parity bit may be transmitted after the data
Transmitting data

- Typically, a FIFO buffer of 1 or more bytes is used for transmission and reception
Transmitting data

- Typically, a FIFO buffer of 1 or more bytes is used for transmission and reception.
- The status of the buffers is probed internally by empty and possibly full status bits.
Transmitting data

- Typically, a FIFO buffer of 1 or more bytes is used for transmission and reception.
- The status of the buffers is probed internally by `empty` and possibly `full` status bits.
- If the read buffer is only 1 byte, `NOT EMPTY` is the same as `DATA READY`.
Transmitting data

- Typically, a FIFO buffer of 1 or more bytes is used for transmission and reception.

- The status of the buffers is probed internally by `empty` and possibly `full` status bits.

- If the read buffer is only 1 byte, `NOT EMPTY` is the same as `DATA READY`.

- If the write buffer is only 1 byte, `NOT EMPTY` is the same as `NOT READY/FULL`. 
Handling Overflow

- What happens if the Master sends data to the slave receive buffer faster than it can be unloaded?
Handling Overflow

- What happens if the Master sends data to the slave receive buffer faster than it can be unloaded?

- To prevent overflow, additional hardware lines can be used to communicate status and perform handshaking.
Handling Overflow

- What happens if the Master sends data to the slave receive buffer faster than it can be unloaded?

- To prevent overflow, additional hardware lines can be used to communicate status and perform handshaking.

- A software handshaking can also be implemented.
Handling Overflow

- What happens if the Master sends data to the slave receive buffer faster than it can be unloaded?
- To prevent overflow, additional hardware lines can be used to communicate status and perform handshaking.
- A software handshaking can also be implemented.
- The stop/start bits, parity, hardware/software hand shaking, and baud rate must be configured on both ends.
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
  - UCSRB
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
  - UCSRB
  - UCSRC
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
  - UCSRB
  - UCSRC
  - UBRRH & UBRRL
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
    - Flags for various errors that might occur during data transmission, e.g. parity error, frame error etc.
  - UCSRB
  - UCSRC
  - UBRRH & UBRRL
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
    - Flags for various errors that might occur during data transmission, e.g. parity error, frame error etc.
  - UCSRB
    - Contains lot of enable bits. eg. different interrupt enable bits and the receiving and transmitting enable bits
  - UCSRC
  - UBRRH & UBRRL
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
    - Flags for various errors that might occur during data transmission, e.g. parity error, frame error etc.
  - UCSRB
    - Contains lot of enable bits. eg. different interrupt enable bits and the receiving and transmitting enable bits
  - UCSRC
    - Set the parity mode, stop bits etc.
  - UBRRH & UBRRL

For Our AVR, the required registers have slightly different names and are in the EXTENDED I/O, meaning, we must access them as memory and not as registers.
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
    - Flags for various errors that might occur during data transmission, e.g. parity error, frame error etc.
  - UCSRB
    - Contains lot of enable bits. eg. different interrupt enable bits and the receiving and transmitting enable bits
  - UCSRC
    - Set the parity mode, stop bits etc.
  - UBRRH & UBRRL
    - The higher byte (UBRRH) and lower byte (UBRRL) is stored for generating the required baud rate
AVR Butterfly UART Registers

- The following set of registers are used to communicate over UART
  - UCSRA
    - Flags for various errors that might occur during data transmission, e.g. parity error, frame error etc.
  - UCSRB
    - Contains lot of enable bits. eg. different interrupt enable bits and the receiving and transmitting enable bits
  - UCSRC
    - Set the parity mode, stop bits etc.
  - UBRRH & UBRRL
    - The higher byte (UBRRH) and lower byte (UBRRL) is stored for generating the required baud rate
- For Our AVR, the required registers have slightly different names and are in the EXTENDED I/O, meaning, we must access them as memory and not as registers
AVR Butterfly Interfacing

RS232 Communications Port

<table>
<thead>
<tr>
<th>Pin #</th>
<th>RS-232</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Carrier Detect (DCD)</td>
</tr>
<tr>
<td>2</td>
<td>RXD</td>
</tr>
<tr>
<td>3</td>
<td>TXD</td>
</tr>
<tr>
<td>4</td>
<td>Data Terminal Ready (DTR)</td>
</tr>
<tr>
<td>5</td>
<td>Ground (GND)</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready (DSR)</td>
</tr>
<tr>
<td>7</td>
<td>Request To Send (RTS)</td>
</tr>
<tr>
<td>8</td>
<td>Clear To Send (CTS)</td>
</tr>
<tr>
<td>9</td>
<td>Ring Indicator (RI)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AVR Butterfly</th>
<th>COM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1 (RXD)</td>
<td>Pin 3</td>
</tr>
<tr>
<td>Pin 2 (TXD)</td>
<td>Pin 2</td>
</tr>
<tr>
<td>Pin 3 (GND)</td>
<td>Pin 5</td>
</tr>
</tbody>
</table>
RealTerm Settings
Interfacing Code

Download interfacing code from your instructor’s website (uart.asm)