Background on UART

- UART = Universal Asynchronous Receive/Transmit
- Also known as the “Serial Port” (RS-232)
  - RS-232 standard is almost 50 years old, so there's a lot of cruft in it
- Asynchronous Interface (no clock used)
- Single Ended (RS-422 has differential signaling)
- Still used because of its relative simplicity
  - Doesn't need complicated driver support
  - Interfaces are fairly cheap and robust
Asynchronous Data Transfer

- RS-232 signal phases
  - Idle
  - Start bit
  - Data
  - Parity
  - Stop bit – channel returns to idle condition
  - Idle or Start next frame
Asynchronous Data Transfer

- What is important is that both the sender and receiver are set to the same parameters
  - Baud rate
  - Parity
  - Number of stop bits
RS-232 Signals

- The “DB-9” configuration contains many different signals, such as:
  - G (5), TxD (3), RxD(2), DTR(4), DSR(6), RTS(7), CTS(8), DCD(1), RI (9)
- Most of these signals were used for modems, and are irrelevant for other uses
- *All we care about are the TX/RX signals*
Asynchronous Data Transfer

- At the high level, there's three key interfaces:
  - Configuration Register(s)
    - Configure interrupts, baud rate, etc.
  - Check the status of the UART
  - Send FIFO
    - Write in data bytes that need to be sent out
    - Finite depth (16 deep for 16550, so make sure there's still room in it)
  - Receive FIFO
    - Read out data bytes that have been received
    - Finite depth (16 deep for 16550)
Altera's UART Implementation

- Hardware-level block diagram:
**Altera's UART Implementation**

- **Software-visible registers:**

  ![Table 1. UART Register Map](image)

<table>
<thead>
<tr>
<th>A2..A0</th>
<th>Register Name</th>
<th>R/W</th>
<th>Description/Register Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>rxdata</td>
<td>RO</td>
<td>RxData</td>
</tr>
<tr>
<td>1</td>
<td>txdata</td>
<td>WO</td>
<td>TxData</td>
</tr>
<tr>
<td>2</td>
<td>status (1)</td>
<td>RW</td>
<td>eop, cts, dcts, -e (2), rrdy, trdy, tmt, toe, roe, brk, fe, pe</td>
</tr>
<tr>
<td>3</td>
<td>control</td>
<td>RW</td>
<td>ieop, rts, idcts, trbk, ie, irrdy, itrdy, itmt, itoe, iroe, ibrk, ife, ipe</td>
</tr>
<tr>
<td>4</td>
<td>divisor</td>
<td>RW</td>
<td>Baud Rate Divisor (optional)</td>
</tr>
<tr>
<td>5</td>
<td>endofpacket</td>
<td>RW</td>
<td>End-packet value</td>
</tr>
</tbody>
</table>
Altera's UART Implementation

- Control register bits:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ipe</td>
<td>Enable interrupt for a parity error</td>
</tr>
<tr>
<td>1</td>
<td>ife</td>
<td>Enable interrupt for a framing error</td>
</tr>
<tr>
<td>2</td>
<td>ibrk</td>
<td>Enable interrupt for a break detect</td>
</tr>
<tr>
<td>3</td>
<td>iroe</td>
<td>Enable interrupt for a receiver overrun error</td>
</tr>
<tr>
<td>4</td>
<td>itoe</td>
<td>Enable interrupt for a transmitter overrun error</td>
</tr>
<tr>
<td>5</td>
<td>itmt</td>
<td>Enable interrupt for a transmitter shift register empty</td>
</tr>
<tr>
<td>6</td>
<td>itrdy</td>
<td>Enable interrupt for a transmission ready</td>
</tr>
<tr>
<td>7</td>
<td>irrty</td>
<td>Enable interrupt for a read ready</td>
</tr>
<tr>
<td>8</td>
<td>ie</td>
<td>Enable interrupt for an exception</td>
</tr>
<tr>
<td>9</td>
<td>trbk</td>
<td>Transmit break</td>
</tr>
<tr>
<td>10</td>
<td>idcts</td>
<td>Enable interrupt for a change in CTS signal</td>
</tr>
<tr>
<td>11</td>
<td>rts</td>
<td>Request to send (RTS) signal</td>
</tr>
<tr>
<td>12</td>
<td>ieop</td>
<td>Enable interrupt for an end of packet encountered</td>
</tr>
</tbody>
</table>
Altera's UART HAL API

/* A simple program that recognizes the characters 't' and 'v'. */
#include <stdio.h>
#include <string.h>
int main ()
{
    char* msg = "Detected the character 't'.\n";
    FILE* fp;
    char prompt = 0;

    fp = fopen ("/dev/uart1", "r+"); //Open file for reading and writing
    if (fp)
    {
        while (prompt != 'v')
        {
            // Loop until we receive a 'v'.
            prompt = getc(fp); // Get a character from the UART.
            if (prompt == 't')
            {
                // Print a message if character is 't'.
                fwrite (msg, strlen (msg), 1, fp);
            }
        }

        fprintf(fp, "Closing the UART file.\n");
        fclose (fp);
    }

    return 0;
}