JTAG UART

A UART (Universally Asynchronous Receiver-Transmitter) core, to allow for communication between a Nios II Terminal and the DE1-SoC Board.

<table>
<tr><td>Device</td><td>JTAG UART</td></tr>
<tr><td>Input/Output</td><td>Both</td></tr>
<tr><td>Address Base</td><td>0xFF201000</td></tr>
<tr><td>Address Map</td><td>
<table>
<tr><th>Address</th><th>R/W</th><th>Description</th></tr>
<tr><td>base</td><td>R/W</td><td>Data Register
31:16 - Number of characters available to read (before this read - see Notes)
15 - read data is valid
7:0 - the data itself</td></tr>
<tr><td>base+4</td><td>R/W</td><td>Control Register
31:16 - Spaces available for writing
10 - AC (has value of 1 if JTAG UART has been accessed by host, cleared by writing 1 to it)
9 - Write interrupt is pending
8 - Read interrupt is pending
1 - Enable write interrupts
0 - Enable read interrupts</td></tr>
</table>
</td></tr>
<tr><td>Initialization</td><td>None</td></tr>
<tr><td>Interrupts</td><td>
<table>
<tr><td>Triggered</td><td>On data-received or able to send (see below)</td></tr>
<tr><td>IRQ Line</td><td>8</td></tr>
<tr><td>Enable</td><td>Set bit 1 and/or 0 in the Control Register for write and read interrupts respectively.</td></tr>
<tr><td>Acknowledge</td><td>Read (write) interrupts acknowledged by reading (writing) to the Data Register</td></tr>
</table>
</td></tr>
<tr><td>Software Setup</td><td>The Monitor Program creates a terminal window automatically, that can be used to send characters to the UART and receive the output from the JTAG UART.</td></tr>
<tr><td>Reference</td><td>Altera JTAG Core Datasheet</td></tr>
</table>

Notes

The Data Register is used for both sending and receiving data — the type of instruction executed (stw or ldw) determines whether you send or receive. Every time you read from this register an 8-bit data byte is ejected from the receive FIFO, and every time you write to it you insert a byte to the send FIFO. Note that reading the data register word will obtain the "Number of Bytes available" in bits [31:16], data valid [15], and will also read the data in bits [7:0] and eject another byte (if any) from the queue. The "Number of Bytes available" returns the utilization of the queue before the current read operation.

Use bit 15 to determine if a read is valid. The "Number of bytes available" must not be used for polling. It returns the state of the queue one cycle earlier than when the bytes are actually available. For example,

reading the data register in the same cycle as a byte is received will result in "1 byte available", but the byte will not be valid, and no bytes will be dequeued from the FIFO.

If you try to send a character when the write FIFO is full, the character will not be sent, and will be lost.

Assembly Example: Sending the character '0' through the JTAG

```
movui r2, 0x30 # ASCII code for 0
movia r7, 0xFF201000 # r7 now contains the base address
stwio r2, 0(r7) # Write the character to the JTAG
```

Assembly Example: Polling the JTAG until valid data has been received

```
POLL:
  movia r7, 0xFF201000 # r7 now contains the base address
  ldwio r2, 0(r7) #  Load from the JTAG
  andi  r3, r2, 0x8000 # Mask other bits
  beq   r3, r0, POLL # If this is 0 (branch true), data is not valid
```

C Example: Sending "Hello World" through the UART

```
#define JTAG_UART_DATA ((volatile int*) 0xFF201000)
#define JTAG_UART_CONTROL ((volatile int*) (0xFF201000+4))

int main()
{
    unsigned char hwld[] = {'H','e','l','l','o',' ','W','o','r','l','d','\0'};
    unsigned char *pOutput;

    pOutput = hwld;
    while(*pOutput) //strings in C are zero terminated
    {
         //if room in output buffer
         if((*JTAG_UART_CONTROL)&0xffff0000  )
         {
            //then write the next character
            *JTAG_UART_DATA = (*pOutput++);
         }
     }
}
```