The LPC2138 has two UARTs called UART0 and UART1. The two are identical except that UART1 has some additional functions that make it easier to control a modem. In this class we will not be using the MODEM controls and the notes below apply to both UARTs.

The interface to the outside world is by way of two pins: the transmit pin is TXD0 and the receive pin is RXD0. On the ARM7 version 5 board the UART0 pins are passed through the FTDI 232 chip to convert the serial interface to USB. The UART0 pins are also brought out directly to CN-2 on Pins 11 (TXD0) and 10 (RXD0). This is the programming port. UART1 pins are brought out directly to Pins 14 (TDX1) and 16 (RDX1) on CN-2.

UART0 has eleven registers which are shown in Figure 1 below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>BIT 7</th>
<th>BIT 6</th>
<th>BIT 5</th>
<th>BIT 4</th>
<th>BIT 3</th>
<th>BIT 2</th>
<th>BIT 1</th>
<th>BIT 0</th>
<th>Access</th>
<th>Reset Value</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>UORBR</td>
<td>Receiver Buffer Register</td>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RO</td>
<td>undefined</td>
<td>0xE000C000 DLAB = 0</td>
</tr>
<tr>
<td>U0THR</td>
<td>Transmit Holding Register</td>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WO</td>
<td>NA</td>
<td>0xE000C000 DLAB = 0</td>
</tr>
<tr>
<td>U0DLL</td>
<td>Divisor Latch LSB</td>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R/W</td>
<td>0x01</td>
<td>0xE000C000 DLAB = 1</td>
</tr>
<tr>
<td>U0DLM</td>
<td>Divisor Latch MSB</td>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>0xE000C004 DLAB = 1</td>
</tr>
<tr>
<td>U0IER</td>
<td>Interrupt Enable Register</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Enable Rx Line Status Interrupt</td>
<td>R/W</td>
<td>0</td>
<td>0xE000C004 DLAB = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U0IIR</td>
<td>Interrupt ID Register</td>
<td>FIFOs Enabled</td>
<td>0</td>
<td>0</td>
<td>IIR3</td>
<td>IIR2</td>
<td>IIR1</td>
<td>IIR0</td>
<td>RO</td>
<td>0x01</td>
<td>0xE000C008</td>
<td></td>
</tr>
<tr>
<td>U0FCR</td>
<td>FIFO Control Register</td>
<td>Rx Trigger</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>Tx FIFO</td>
<td>Rx FIFO</td>
<td>WO</td>
<td>0</td>
<td>0xE000C008</td>
<td></td>
</tr>
<tr>
<td>U0LCR</td>
<td>Line Control Register</td>
<td>DLAB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Word Length</td>
<td>R/W</td>
<td>0</td>
<td>0xE000C00C</td>
<td></td>
</tr>
<tr>
<td>U0LSR</td>
<td>Line Status Register</td>
<td>RXF</td>
<td>Error</td>
<td>TXE</td>
<td>RX parity</td>
<td>RX parity</td>
<td>Number of Stop Bits</td>
<td>Word Length</td>
<td>RO</td>
<td>0x60</td>
<td>0xE000C014</td>
<td></td>
</tr>
<tr>
<td>U0SCR</td>
<td>Scratch Pad Register</td>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>0xE000C01C</td>
</tr>
<tr>
<td>U0TER</td>
<td>Transmit Enable</td>
<td>TxEn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>0xE000C020</td>
</tr>
</tbody>
</table>

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

**Figure 1**
UART0 register definitions

To use the UART for simple transmission and reception we need to set up the prescaler divider registers (U0DLM and U0DLL), the line control register (U0LCR), pclk (VPBDIV), and the pin select register to enable the UART pins for output and input (PINSEL0). The remaining register reset states allow the UART to run. To transmit data you write the U0 transmit holding register
(U0THR) and to receive data you read it from the U0 Receive Buffer Register (U0RBR). The Line Status Register can be used to determine if the UART has a character to read or if its transmit buffer is empty. Since the processor can read and write much faster than the UART can transmit data, you must always check the bits in these registers to determine the UART status before reading or writing.

**Line Control Register**
The Line Control Register U0LCR, is used to determine such things as the number of stop bits, even or odd parity, etc.

| Table 68: UART0 Line Control Register Bit Descriptions (U0LCR - 0xEO00C00C) |
|-----------------|-------------------------------------------------|
| **U0LCR** | **Function** | **Description** | **Reset Value** |
| 1:0   | Word Length Select | 00: 5 bit character length  
01: 6 bit character length  
10: 7 bit character length  
11: 8 bit character length | 0 |
| 2     | Stop Bit Select | 0: 1 stop bit  
1: 2 stop bits (1.5 if U0LCR[1.0]=00) | 0 |
| 3     | Parity Enable | 0: Disable parity generation and checking  
1: Enable parity generation and checking | 0 |
| 5:4   | Parity Select | 00: Odd parity  
01: Even parity  
10: Forced ‘1’ stick parity  
11: Forced ‘0’ stick parity | 0 |
| 6     | Break Control | 0: Disable break transmission  
1: Enable break transmission. Output pin UART0 TxD is forced to logic 0 when U0LCR6 is active high | 0 |
| 7     | Divisor Latch Access Bit | 0: Disable access to Divisor Latches  
1: Enable access to Divisor Latches | 0 |

**Figure 2**
Line Control Register
For our applications in this class we will have an 8-bit character length, 1 stop bit, disabled parity checking, disabled break control, and we will enable access to the Divisor latches. For example

U0LCR = 0x83; //8-bit, 1 stop bit, no parity, enable divisor latch

**Pin Select Register**
To enable UART0 I/O pins bits 3, 2, 1, and 0 should be set to 0101. (UART1 needs bits 19, 18, 17, and 16 set to 0101.

PINSEL0 = 0x5;

**Baud Rate Prescale Registers**
Both UARTs have their own baud rate generators. The input to these generators is the peripheral clock (pclk). The baud rate generator contains a 16-bit prescaler which divides pclk. The UART clock must run at 16 times the baud rate. For example, if we have a 60MHz crystal clock and we set

VPBDIV = 0x2; //make pclk = 30 MHz

then, for a baud rate of 9600 baud we want a divisor = pclk/(16 x 9600) = 195.3125. Rounding this to 195 = 0xC3 gives a true baud rate of 9615.

U0DLM = 0;  
U0DLL = 0xC3;
Line Status Register

The line status register bit definitions are shown in Figure 3 below.

Table 67: UART0 Line Status Register Bit Descriptions (U0LSR - 0x8000C014, Read Only)

<table>
<thead>
<tr>
<th>U0LSR</th>
<th>Function</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Receiver Data Ready (RDR)</td>
<td>0: U0RBR is empty. 1: U0RBR contains valid data. U0RBR is set when the U0RBR holds an unread character and is cleared when the UART0 RBR FIFO is empty.</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Overrun Error (OE)</td>
<td>0: Overrun error status is inactive. 1: Overrun error status is active. The overrun error condition is set as soon as it occurs. An U0LSR read clears U0LSR1. U0LSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost.</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Parity Error (PE)</td>
<td>0: Parity error status is inactive. 1: Parity error status is active. When the parity bit of a received character is in the wrong state, a parity error occurs. An U0LSR read clears U0LSR2. Time of parity error detection is dependent on U0FCRD. A parity error is associated with the character at the top of the UART0 RBR FIFO.</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Framing Error (FE)</td>
<td>0: Framing error status is inactive. 1: Framing error status is active. When the stop bit of a received character is a logic 0, a framing error occurs. An U0LSR read clears U0LSR3. The time of the framing error detection is dependent on U0FCRD. A framing error is associated with the character at the top of the UART0 RBR FIFO. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error.</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Break Interrupt (BI)</td>
<td>0: Break interrupt status is inactive. 1: Break interrupt status is active. When Rx0 is held in the spacing state (all 0s) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until Rx0 goes to marking state (all 1s). An U0LSR read clears this status bit. The time of break detection is dependent on U0FCRD. The break interrupt is associated with the character at the top of the UART0 RBR FIFO.</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Transmitter Holding Register Empty (THRE)</td>
<td>0: U0THR contains valid data. 1: U0THR is empty. THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Transmitter Empty (TEMT)</td>
<td>0: U0THR and/or the U0TSR contains valid data. TEMT is set when both U0THR and U0TSR are empty. TEMT is cleared when either the U0TSR or the U0THR contain valid data.</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Error in Rx FIFO (E0FE)</td>
<td>0: U0RBR contains no UART0 Rx errors or U0FCRD=0. 1: UART0 RBR contains at least one UART0 Rx error. U0LSR7 is set when a character with a Rx error such as framing error, parity error or break interrupt is loaded into the USRBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART0 FIFO.</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3
Line Status Register

Bit 0 in this register may be tested to determine if a character has been received and is in the buffer. Bit 5 can be used to determine if the transmit holding register is empty. To transmit a character we can poll bit 5 as shown below:

```c
while (!((U0LSR & 0x20)));  //Test bit 5
   U0THR = ch;               //When bit 5 = 1, load data to transmit ch
```
Likewise, to receive a character in a polling mode:

```c
while (!((U0LSR & 0x01)));  //Test bit 0
   return (U0RBR);           //When bit 0 = 1, get data from buffer
```

Example 1
Set up UART0 for 9600 baud, no parity, 1 stop bit, and 8-bits of data to transmit the message "Hello Mom!". Use the simulator to view the results.
//UARTTest1
// Transmits "Hello Mom!" to Serial Window 1 in the simulator
// at 9600 baud.
#include<lpc213x.h>
char msg[] = "Hello Mom!";
int putchar(int ch);
int getchar(void);
int main()
{char c;
 int i;
 VPBDIV = 0x2;
 PINSEL0 = 0x5; // Enable RxD0 and TxD0
 U0LCR = 0x00000083; // 8 bits, no Parity, 1 Stop bit
 U0DLL = 0x000000C3; // 9600 Baud Rate @ 30MHz VPB Clock
 U0LCR = 0x00000003; // DLAB = 0
 i = 0;
 while((c = msg[i]) != 0)
 {putchar(c);
  i++;
 }
 return 0;
}
int putchar(int ch) // Write character to Serial Port
{if (ch == '\n')
 {while (!((U0LSR & 0x20));
  U0THR = 0x0D; //if new line send cr
 }
 while (!((U0LSR & 0x20));
 return (U0THR = ch); //Otherwise send char
}
int getchar (void) /* Read character from Serial Port */
{while (!((U0LSR & 0x01));
 return (U0RBR);
}

Figure 4
Transmits "Hello Mom!" on UART0. In the simulator you can see the results in Serial Window 1.