Industrial Reference Design Platform
RS-232 Interface
Developed by the TSC Americas

Release 1.0
LPC2468 UARTs
Functional Overview

- There are 4 UARTS available on the LPC2468

- UART functionality
  - 16 byte Receive and Transmit FIFOs
  - Register locations conform to industry standard 16C550 UART
  - Receiver FIFO trigger points at 1, 4, 8, and 14 bytes
  - Fractional divider for baud rate control, auto-baud capabilities and mechanism that enables software flow control implementation
  - UART1 allows for implementation of either software or hardware flow control
  - UART3 includes an IrDA mode to support infrared communication
  - Maximum possible speed of the UART – 4.5Mb/s
LPC2468 UARTs 0, 2, and 3
Block Diagram
LPC2468 UART 1
Block Diagram

Modem Interface signals
- CTS
- DSR
- RI
- DCD
- DTR
- RTS

LSR
Line Status Register

LCR
Line Control Register

SCR
Scratch Pad Register

MODEM

MSR

MCR

VPB Bus Interface

Tx
(Transmitter Block)

THR
Tx Holding Register

TSR
Tx Shift Register

BRG
Baud Rate Generator Block

DLL

Divisor Latch LSB

DLM

Divisor Latch MSB

FDR
Fractional Divisor register

Rx
(Receiver Block)

RBR
Rx Buffer Register

RSR
Rx Shift Register

FCR
FIFO Control Register

Serial Output Pin

Serial Input Pin
LPC2468 UARTs
Baud Rate Generators

Table 340: UARTn Divisor Latch LSB Register (U0DLL - address 0xE000 C000, U2DLL - 0xE007 8000, U3DLL - 0xE007 C000 when DLAB = 1) bit description

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>DLLSB</td>
<td>The UARTn Divisor Latch LSB Register, along with the UnDLM register, determines the baud rate of the UARTn.</td>
<td>0x01</td>
</tr>
</tbody>
</table>

Table 341: UARTn Divisor Latch MSB Register (U0DLM - address 0xE000 C004, U2DLM - 0xE007 8004, U3DLM - 0xE007 C004 when DLAB = 1) bit description

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>DLMSB</td>
<td>The UARTn Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UARTn.</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Note: For DLLSB, a 0x00 value is treated like a 0x01 value, as division by zero is not allowed.
LPC2468 UARTs
Fractional Baud Rate Generator (Available on Enhanced UARTs)

- Accurate Baud Rate Generation from a wide range of Crystal Frequencies

\[ \text{UART}_{n\text{baudrate}} = \frac{PCLK}{16 \times UnDL} \times \frac{MULVAL}{(MULVAL + DIVADDVAL)} \]

UnDL is value determined by the UnDLM and UnDLL registers
(UnDL = 256 \times UnDLM + UnDLL)

DIVADDVAL and MULVAL are UART\text{n}
fractional baud-rate generator specific parameters
specified in the Fractional Divisor (UnFDR) register

PCLK is the peripheral clock frequency set for the specific
UART in the PCLKSELn register (Peripheral clock divisors can
be set differently for individual peripherals in the LPC23/24xx)
LPC2468 UARTs
Fractional Baud Rate Generator Registers

Table 352: UARTn Fractional Divider Register (U0FDR - address 0xE000 C028, U2FDR - 0xE007 8028, U3FDR - 0xE007 C028) bit description

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Value</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:0</td>
<td>DIVADDVAL</td>
<td>0</td>
<td>Baud-rate generation pre-scaler divisor value. If this field is 0, fractional baud-rate generator will not impact the UARTn baudrate.</td>
<td>0</td>
</tr>
<tr>
<td>7:4</td>
<td>MULVAL</td>
<td>1</td>
<td>Baud-rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud-rate generator is used or not.</td>
<td>1</td>
</tr>
<tr>
<td>31:8</td>
<td>-</td>
<td>NA</td>
<td>Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.</td>
<td>0</td>
</tr>
</tbody>
</table>

The value of MULVAL and DIVADDVAL should comply to the following conditions:
- $0 < \text{MULVAL} \leq 15$
- $0 \leq \text{DIVADDVAL} \leq 15$
# LPC2000 Family UART

## Fractional Baud Rate Calculator Tool

![Baudrate Calculator](calc.png)

### Please enter the following two parameters:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPC2000 UART Baudrate Calculator</td>
<td>UART clock [Hz]</td>
<td>120000000</td>
<td>Requested UART baudrate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Maximum allowed relative error [%]</td>
</tr>
</tbody>
</table>

### Best available fit for standard UART

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART baudrate</td>
<td>107142.86</td>
<td>Relative error</td>
<td>-6.99%</td>
</tr>
<tr>
<td>UDL [dec]</td>
<td>7</td>
<td>UDLM [dec]</td>
<td>0</td>
</tr>
<tr>
<td>UDLL [dec]</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Best available fit for enhanced UART

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART baudrate</td>
<td>115384.62</td>
<td>Relative error</td>
<td>0.16%</td>
</tr>
<tr>
<td>UDL [dec]</td>
<td>6</td>
<td>UDLM [dec]</td>
<td>0</td>
</tr>
<tr>
<td>UDLL [dec]</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DivAddVal [dec]</td>
<td>1</td>
<td>MuVal [dec]</td>
<td>12</td>
</tr>
</tbody>
</table>

**Terms of Use:** [http://www.nxp.com/terms](http://www.nxp.com/terms)

IRD RS-232 Interface
Implementation

- UART0 and UART1 provide RS-232 interfaces with DB9 connectors
- UART2 is pinned out to JP1-7(Tx) and JP1-9 (Rx)
- UART3 is pinned out to expansion header JP3-4(Tx) and JP3-1(Rx)
- UART0 also functions as a programming interface for ISP
IRD RS-232 Interface Schematic

DTE/DCE Selection

DTE/DCE Flexibility

* = Default Configuration

<table>
<thead>
<tr>
<th></th>
<th>DCE</th>
<th>DTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP9</td>
<td>1-2 *</td>
<td>1-3</td>
</tr>
<tr>
<td></td>
<td>3-4 *</td>
<td>2-4</td>
</tr>
<tr>
<td>JP12</td>
<td>1-3</td>
<td>1-2 *</td>
</tr>
<tr>
<td></td>
<td>2-4</td>
<td>3-4 *</td>
</tr>
</tbody>
</table>
IRD RS-232 Interface Schematic

In System Programming (ISP)

<table>
<thead>
<tr>
<th>ISP</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>J10</td>
<td>1-2</td>
</tr>
<tr>
<td>J8</td>
<td>1-2</td>
</tr>
<tr>
<td>J9</td>
<td>1-2,3-4</td>
</tr>
</tbody>
</table>

UART 0 Used for In System Programming
IRD RS-232 Interface Software
UART Initialization Flowchart

1. Declare Local Variables
2. Init UART 0 or 1?
   - UART0
     1. Init Local Variables
     2. Set Up I/O Registers
     3. Init Baud Rate Registers
     4. Init Serial Mode
     5. Init Interrupt mode
     6. Init FIFO mode
   - UART1
     1. Init Local Variables
     2. Set Up I/O Registers
     3. Init Baud Rate Registers
     4. Init Serial Mode
     5. Init Interrupt mode
     6. Init FIFO mode
IRD RS-232 Interface Software
UART Write Function Flowchart

UART0

UART 0 or 1?

UART1

Is U0THR Empty?

Load U0THR with Byte to Transmit

Is U1THR Empty?

Load U1THR with Byte to Transmit
IRD RS-232 Interface Software
UART Read Function Flowchart

UART0

UART 0 or 1?

Is U0 RDR Bit Set?

Get Received Byte From U0RBR Register

UART1

Is U1 RDR Bit Set?

Get Received Byte From U1RBR Register
void Ser_Init (void)
{
    CPU_INT16U divisor;  /* Baud rate divisor */
    CPU_INT08U divlo;
    CPU_INT08U divhi;
    CPU_INT32U pclk_freq;
    CPU_INT32U pinsel;
    #if (SER_COMM_SEL == SER_UART_0)
        pclk_freq = BSP_CPU_PclkFreq(PCLK_UART0); /* Get the CPU clock frequency */
        divisor = (CPU_INT16U)(((2 * pclk_freq / 16 / 115200) + 1) / 2);
        divlo = divisor & 0x00FF;  /* Split divisor into L and H bytes */
        divhi = (CPU_INT08U)(divisor >> 8) & 0x00FF;
        pinsel = PINSEL0; /* Configure P0.2 & P0.3 for UART0 */
        pinsel &= 0xFFFFFFFF0F;
        pinsel |= 0x00000050;
        PINSEL0 = pinsel;
        U0LCR = DEF_BIT_07;  /* Set divisor access bit */
        U0DLL = divlo;       /* Load divisor */
        U0DLM = divhi;
        U0LCR = DEF_BIT_00 | DEF_BIT_01;  /* 8 Bits, 1 Stop, No Parity */
        U0IER = 0x00;        /* Disable both Rx and Tx interrupts */
        U0FCR = DEF_BIT_00 | DEF_BIT_01 | DEF_BIT_02; /* Enable FIFO, flush Rx & Tx */
    #endif
}
#if (SER_COMM_SEL == SER_UART_1)

    pclk_freq = BSP_CPU_PclkFreq(PCLK_UART1); /* Compute divisor for desired baud rate */
    /* Get the CPU clock frequency. */
    divisor   = (CPU_INT16U)(((2 * pclk_freq / 16 / 115200) + 1) / 2);
    divlo     = divisor & 0x00FF;           /* Split divisor into L and H bytes */
    divhi     = (CPU_INT08U)(divisor >> 8) & 0x00FF;
    pinsel    = PINSEL7;                   /* Configure P3.16 & P3.17 for UART1 */
    pinsel &= 0xFFFFFFFF0;
    pinsel |= 0x0000000F;
    PINSEL7   = pinsel;
    U1LCR     = DEF_BIT_07;             /* Set divisor access bit */
    U1DLL     = divlo;                  /* Load divisor */
    U1DLM     = divhi;
    U1LCR     = DEF_BIT_00 | DEF_BIT_01; /* 8 Bits, 1 Stop, No Parity */
    U1IER     = 0x00;                   /* Disable both Rx and Tx interrupts */
    U1FCR     = DEF_BIT_00 | DEF_BIT_01 | DEF_BIT_02; /* Enable FIFO, flush Rx & Tx */
#endif
}
IRD RS-232 Interface Software
UART Write Byte and Write String Functions Code

```c
void Ser_WrByte (CPU_CHAR tx_byte)
{
    #if (SER_COMM_SEL == SER_UART_0)
        while ((U0LSR & DEF_BIT_05) == 0) {
            ;
        }
        U0THR = tx_byte;
    #endif

    #if (SER_COMM_SEL == SER_UART_1)
        while ((U1LSR & DEF_BIT_05) == 0) {
            ;
        }
        U1THR = tx_byte;
    #endif
}

void Ser_WrStr (CPU_CHAR *tx_str)
{
    while (*tx_str != 0) {
        Ser_WrByte(*tx_str);
        *tx_str++;
    }
}
```
IRD RS-232 Interface Software
UART Read Byte Function Code

```
CPU_INT08U  Ser_RdByte (void)
{
  CPU_INT08U  rx_byte;

#if (SER_COMM_SEL == SER_UART_0)
  while (((U0LSR & DEF_BIT_00) == 0) {
    OSTimeDly(1);
  }
  rx_byte = (CPU_INT08U)(U0RBR);  /* Remove the data from the holding register*/
  return (rx_byte);
#endif

#if (SER_COMM_SEL == SER_UART_1)
  while (((U1LSR & DEF_BIT_00) == 0) {
    OSTimeDly(1);
  }
  rx_byte = (CPU_INT08U)(U1RBR);  /* Remove the data from the holding register */
  return (rx_byte);
#endif
}
```
void Ser_RdStr (CPU_CHAR *rx_str,
    CPU_INT32U   len)
{
    CPU_CHAR input;
    CPU_CHAR input_ix;

    input_ix = 0;
    rx_str[0] = 0;

    while (1)
    {
        input = Ser_RdByte();
        if (((input == 'r') ||
            (input == '\n')))
        {
            Ser_WrByte('\n');
            rx_str[input_ix] = 0;
            return;
        }
        if (Str_IsPrint(input)) {
            Ser_WrByte(input);
            rx_str[input_ix] = input;
            input_ix++;
            if (input_ix >= len) {
                input_ix = len;
            }
        }
    }
}
void Ser_Printf (CPU_CHAR *format, ...)
{
    static CPU_CHAR buffer[80 + 1];
    va_list vArgs;

    va_start(vArgs, format);
    vsprintf((char *)buffer, (char const *)format, vArgs);
    va_end(vArgs);

    Ser_WrStr((CPU_CHAR*) buffer);
}
LPC2000 Baud Rate Calculator Exercise
Invoking the Program

- Download and run the file: lpc2000.uart.baudrate.calculator.xls

- You should see a window similar to the one shown at the right

- This shows the suggested register values and relative errors for the standard UART and the enhanced UART (with the fractional baud rate generator)
LPC2000 Baud Rate Calculator Exercise

Example 1

- Try entering some different values for UART CLK, for example, use 16Mhz
  - Notice that the results for a standard UART shows a high error rate of 3.55%
  - The enhanced UART with the fractional baud rate generator only has an error of 0.16%

![LPC2000 UART Baudrate Calculator](image-url)
LPC2000 Baud Rate Calculator Exercise

Example 2

- Enter 14.7456MHz for UART CLK
  - This is a standard baud rate multiple, so the error is 0%, even for the standard UART
  - However, this also shows that if you didn’t have the enhanced UART, you may need to compromise performance to get accurate baud rates
IRD RS-232 Communications Exercise

Set up

- Connect an RS-232 cable from your PC to the IRD board UART0 connector
- Make sure ISP jumpers JP8 and JP10 are NOT connected.
- Start a terminal program (i.e. HyperTerminal or TeraTerm)
  - Set it up for 115,200 baud, 8, N,1, and no flow control
- Reset the board and see if you get a command prompt ‘ > ‘ on the terminal program display
- Try issuing some of the shell commands listed on the next slide.
  - Typing “Sh_help” will also give you a list of the commands
## IRD RS-232 Communications Exercise

### Shell Command Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sh_help</strong></td>
<td>List the supported commands.</td>
</tr>
<tr>
<td><strong>FS_ls</strong></td>
<td>List information about files in the current directory</td>
</tr>
<tr>
<td><strong>FS_cd [dir]</strong></td>
<td>Change the current directory to [dir]</td>
</tr>
<tr>
<td><strong>FS_pwd</strong></td>
<td>Print the current working directory</td>
</tr>
<tr>
<td><strong>FS_cp</strong></td>
<td>Copy [source] to [dest]</td>
</tr>
<tr>
<td><strong>FS_mv</strong></td>
<td>Rename [source] to [dest] or move [source] to [dir]</td>
</tr>
<tr>
<td><strong>FS_rm</strong></td>
<td>Remove [file]</td>
</tr>
<tr>
<td><strong>FS_rmdir [dir]</strong></td>
<td>Remove [dir], if it is empty</td>
</tr>
<tr>
<td><strong>FS_mkdir [dir]</strong></td>
<td>Create [dir], if it does not already exist</td>
</tr>
<tr>
<td><strong>FS_cat [file]</strong></td>
<td>Concatenate [file] contents to standard output</td>
</tr>
<tr>
<td><strong>FS_od [format] [file]</strong></td>
<td>Dump [file] to standard output in specified format.</td>
</tr>
<tr>
<td><strong>FS_vol</strong></td>
<td>List information about volumes</td>
</tr>
</tbody>
</table>