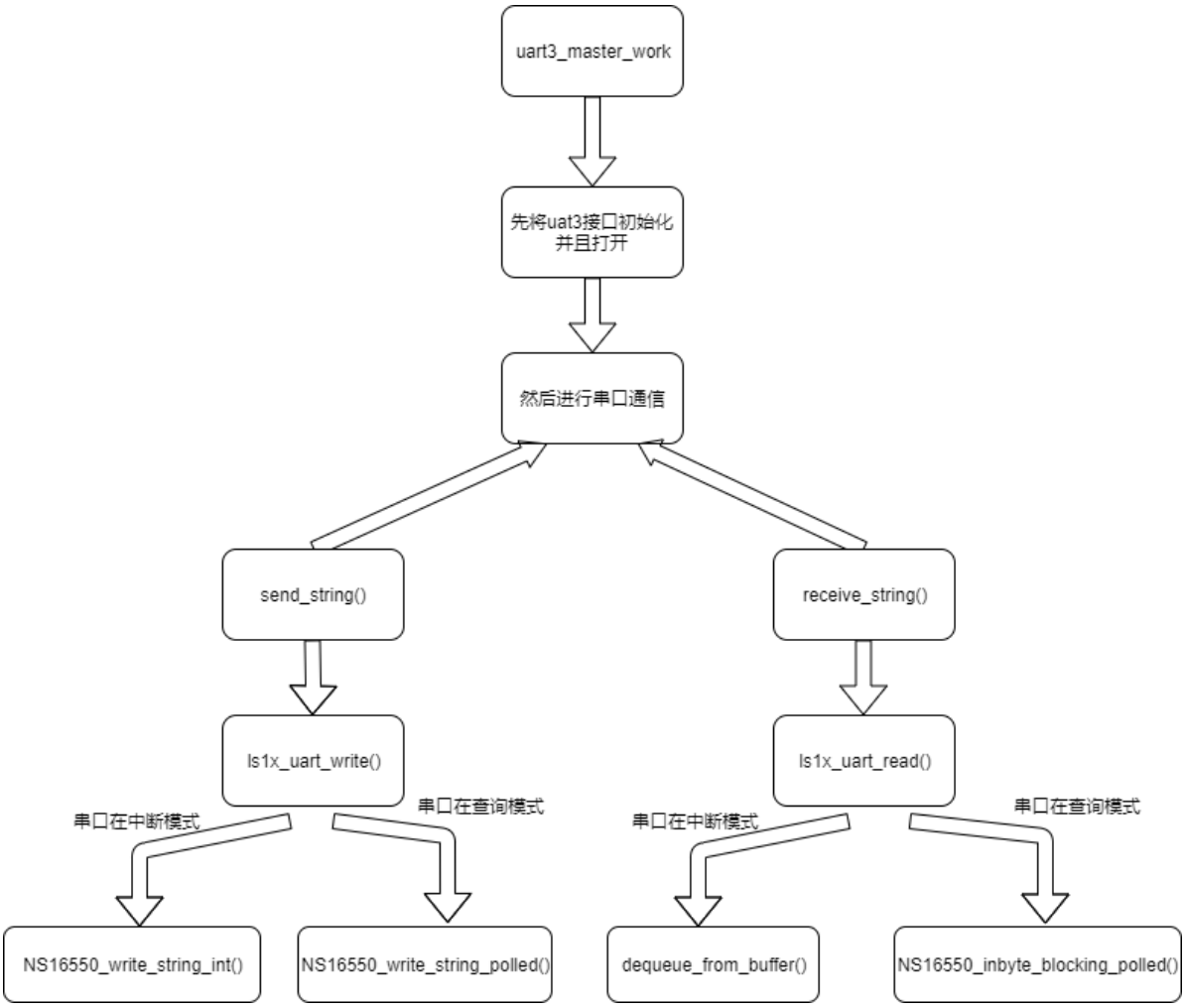
8 串口通信

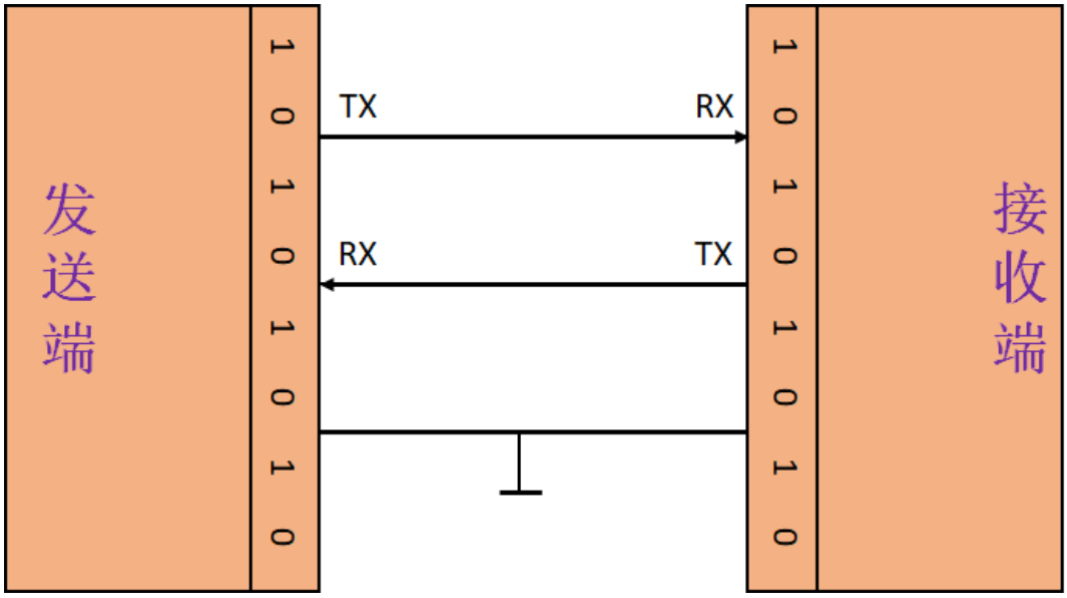
8.1 实验目的

- 了解简单串口通信的过程
- 我们将提供一个demo模板，其功能已经能够实现串口的通信
- 同学们需要理解demo中uart3_master_work()里面的功能
- 上机实验则需要进行演示出相关现象

8.2 过程调用关系



通信两端过程示意图



8.3 主要函数讲解

8.3.1 ls1x_uart_init(uart, arg)

当arg设置为0或NULL时，串口设置为默认模式115200,8N1

```
pUART->BusClock = LS1x_BUS_FREQUENCY(CPU_XTAL_FREQUENCY);
    if (arg != NULL)
    {
        pUART->BaudRate = (unsigned int)arg;
    }

divisor = NS16550_BAUD_DIV(pUART->BusClock, pUART->BaudRate);
```

1、清除division latch，清除所有中断使能

```
NS16550_set_r(pUART->CtrlPort, NS16550_LINE_CONTROL, 0);
//将这个NS16550_LINE_CONTROL寄存器设置为0（字符值）
NS16550_set_interrupt(pUART, NS16550_DISABLE_ALL_INTR);
//清除所有中断的信号
```

```
typedef struct uart_reg
{
    volatile unsigned char reg;
} uartReg;

static unsigned char NS16550_get_r(unsigned CtrlPort, unsigned char RegNum)
{
    struct uart_reg *p = (struct uart_reg *)CtrlPort;
    unsigned char ch = p[RegNum].reg;

    return ch;
}

static void NS16550_set_r(unsigned CtrlPort, unsigned char RegNum, unsigned char
ch)
{
```

```
    struct uart_reg *p = (struct uart_reg *)CtrlPort;
    p[RegNum].reg = ch;
}
//通过这几个函数完成对底部寄存器的访问及设置
static void NS16550_set_interrupt(NS16550_t *pUART, int mask)
{
#if (NS16550_SUPPORT_INT)
    NS16550_set_r(pUART->CtrlPort, NS16550_INTERRUPT_ENABLE, mask);
#else
    NS16550_set_r(pUART->CtrlPort, NS16550_INTERRUPT_ENABLE, 0);
#endif
}
//这个函数可以屏蔽指定的中断
```

2、让这个division latch寄存器置为有效位并设置波特率

3、清除division latch并将字符大小设置为8位，具有一个停止位，无奇偶校验。

4、启用读写区，清除所有中断使能

8.3.2 ls1x_uart_open(uart, arg)

1、设置初始波特率

2、初始化读写缓存区

```
initialize_buffer(&pUART->RxData);
initialize_buffer(&pUART->TxData);
```

3、如果使用中断的标识符（bIntrrupt）为1，则安装中断程序，设置中断寄存器

8.3.3 send_string

该函数的功能可以做到从机(实验课上所拿到的板子)向主机(所连接的电脑)发送信息

8.3.3.1 ls1x_uart_write(uart, buf, size, arg)

如果不在中断情况下，那就是在查询状态下

```
/*
 * 向串口写数据(发送)
 * 参数:    dev     见上面定义的 UART 设备
 *          buf     类型 char *, 用于存放待发送数据的缓冲区
 *          size    类型 int, 待发送的字节数, 长度不超过 buf 的容量
 *          arg     总是 0 或 NULL
 *
 * 返回:    发送的字节数
 *
 * 说明:    串口工作在中断模式: 写操作总是写的内部数据发送缓冲区
 *          串口工作在查询模式: 写操作直接对串口设备进行写
 */
int NS16550_write(void *dev, void *buf, int size, void *arg);
```

- 如果串口工作在中断模式

```
count = NS16550_write_string_int(pUART, (char *)buf, size);
```

- transmit buffer最大有16个字长，如果输出的字符串超过16个字长，则需要再将余下的存入transmit cached buffer。
- 在存入transmit buffer之后，要将中断位置NS16550_ENABLE_ALL_INTR
- 在存入transmit cached buffer时，需要先关中断

  mips_interrupt_disable();

  - 之后进入enqueue_to_buffer(pUART, &pUART->TxData, buf + sent, len - sent)

    这个函数会将余下的字符，存入TxData段中
  - 再开启中断mips_interrupt_enable();

- 如果串口在查询状态

```
count = NS16550_write_string_polled(pUART, (char *)buf, size);
```

  - 先保存中断掩码irq_mask = NS16550_get_r(pUART->CtrlPort, NS16550_INTERRUPT_ENABLE)
  - 再关闭所有中断
  - 关闭该部分的中断（mips_interrupt_disable();）
  - 然后输入字符串
  - 再开启部分中断（mips_interrupt_enable();）
  - 最后还原端口中断掩码

8.3.4 receive_string

该函数的功能可以做到主机(所连接的电脑)向从机(实验课上所拿到的板子)发送信息。

注意：receive主机发送的字符串，写入缓冲区，缓冲区是一个队（先进先出），每次都会弹出缓冲区的前size个字符进行返回

8.3.4.1 ls1x_uart_read(uart, buf, size, arg)

```
/*
 * 从串口读数据(接收)
 * 参数:    dev     见上面定义的 UART 设备
 *          buf     类型 char *, 用于存放读取数据的缓冲区
 *          size    类型 int, 待读取的字节数, 长度不能超过 buf 的容量
 *          arg     类型 int.
 *                  如果串口工作在中断模式:
 *                    >0:   该值用作读操作的超时等待毫秒数
 *                    =0:   读操作立即返回
 *                  如果串口工作在查询模式:
 *                    !=0:  读操作工作在阻塞模式, 直到读取 size 个字节才返回
 *                    =0:   读操作立即返回
 *
 * 返回:    读取的字节数
 *
 * 说明:    串口工作在中断模式: 读操作总是读的内部数据接收缓冲区
 *          串口工作在查询模式: 读操作直接对串口设备进行读
 */
int NS16550_read(void *dev, void *buf, int size, void *arg);
```

- 如果是中断模式
  - 先mips_interrupt_disable();
  - 之后从RxData中取出所读取字符
  - 再mips_interrupt_enable();

如果出现了time out read的情况，则会进行一个while循环进行timeout时常的等待

等到之后，会继续上述操作

- 如果是查询模式
  - 如果arg等于0
    - 则只会读取一个字符
  - 如果不等于0
    - ```
      for (i=0; i<size; i++)
              {
                  val = NS16550_inbyte_blocking_polled(pUART);
                  pchbuf[i] = (char)val;
              }
              //则会通过这样从NS16550_RECEIVE_BUFFER读出size个字节
      ```

8.4 上机注意事项

- 先根据教程连接板子，跑通所给代码，初步完成通信。
- 再根据相关实验要求，完成相关演示。
      ```