Fuart User Guide

Version	Date	Author	Description of Change
V1.0	2012-09-05	Robert	Initial Version For Fast Uart
V1.1	2014-01-20	Robert	Add Fuart Interrupt Service Routine Example
V1.2	2014-08-21	Robert	Add RTS/CTS Description
V1.3	2014-10-09	Sean	Add Baud Rate Noting

Fast Uart 模块

1.1 简介

FUART 支持 1200bsp~3,000,000bps 波特率，(如果是 RC48MHz，波特率支持范围减半，可能有一定的偏差)8/7/6/5 位数据，1/2 位停止位，奇偶校验以及内部深度为 4 的 FIFO，具体的 Feature List 如下:

- 全双工传输。
- 波特率可调，96M 时钟频率下支持 1200bps~3Mbps。
- 8 倍过采样。
- 数据位宽支持 5~8bit。
- 停止位支持 1~2bit。
- 支持奇偶校验。
- 支持帧格式错误检查、溢出错误检查和校验位错误检查。
- 支持接收中断和发送中断以及中断屏蔽。
- 支持接收 FIFO 和发送 FIFO，可使用内部深度为 4 字节的 FIFO。

1.2 接口函数

```
SDWORD FuartInit(DWORD BaudRate,BYTE DatumBits,BYTE Parity,BYTE StopBits);
SDWORD FuartIOctl(SDWORD Cmd,DWORD Opt);
SDWORD FuartRecv(BYTE* Buf,DWORD BufLen,DWORD Wait);
SDWORD FuartSend(BYTE* Buf,DWORD BufLen);
void GpioFuartRxIoConfig(unsigned char ModeSel);
void GpioFuartTxIoConfig(unsigned char ModeSel);
```

1.3 接口函数描述

1.3.1 FuartInit

原型：

```
SDWORD FuartInit(DWORD BaudRate,BYTE DatumBits,BYTE Parity,BYTE StopBits);
```

参数：

BaudRate：波特率，96MHz 主频下，取值[1200:3000000]，48MHz 最大 1,500,000

DatumBits: 数据位数，取值{5,6,7,8}

Parity： 奇偶校验，取值{0=无校验，1=奇校验，2=偶校验}

StopBits： 停止位, 取值{1=1bit 停止位，2=2bit 停止位}

描述：

初始化 FUART 硬件模块

返回值：

0: 成功

EINVAL:参数不正确

1.3.3 FuartIOctl

原型：

SDWORD FuartIOctl(SDWORD Cmd,DWORD Opt);

参数：

Cmd: I/O 控制命令, (以 SDK 中 uart.h 命令为准)取值{

```
UART_IOCTL_BAUDRATE_GET,          //获得当前波特率
UART_IOCTL_BAUDRATE_SET,          //设置波特率

UART_IOCTL_RXFIFO_TRGRDEP_GET, //获得 RXFIFO 触发深度
UART_IOCTL_RXFIFO_TRGRDEP_SET, //设置 RXFIFO 触发深度
UART_IOCTL_RXFIFO_CLR,            //清空 RXFIFO
UART_IOCTL_DISENRX_SET,           //disable/enable RX, Opt=0，
disable;Opt=1，enable，下同。
UART_IOCTL_DISENTX_SET,           //disable/enable TX
UART_IOCTL_RXINT_SET,             //disable/enable RX interrupt
UART_IOCTL_TXINT_SET,             //disable/enable RX interrupt
UART_IOCTL_RXINT_CLR,             //clean RX interrupt status
UART_IOCTL_TXINT_CLR,             // clean TX interrupt status
UART_IOCTL_RXSTAT_GET,            // get RX status
UART_IOCTL_TXSTAT_GET,            //get TX status
}
```

描述：

控制 FUART 各种 I/O 行为。

返回值：

0 或其它值: 成功

EINVAL: 参数不正确

ENOSYS: 无此命令

1.3.4 FuartRecv

原型：

SDWORD FuartRecv(BYTE* Buf,DWORD BufLen,DWORD Wait);

参数：

Buf: 数据接收缓冲区

BufLen：缓冲区长度

Wait：等待时间，时间单位为在某波特率下，传输一个 BYTE 需要的时间

描述：

FUART 数据接收

返回值：

>0: 成功(可能部份接收,即 RecvData>0&&RecvData<=BufLen)

ETIME: 超时失败

1.3.4 FuartSend

原型：

SDWORD FuartSend(BYTE* Buf,DWORD BufLen);

参数：

Buf: 数据发送缓冲区

BufLen: 缓冲区长度

描述：

FUART 数据发送

返回值：

>0: 成功

1.3.5 RX/TX IO 复用

原型：

void GpioFuartRxIoConfig(unsigned char ModeSel);
void GpioFuartTxIoConfig(unsigned char ModeSel);

参数：

ModeSel: GPIO RX/TX 复用关系索引,RX/TX 两两可任意组合
RX 取值{0=GPA13,1=GPB6,2=GPC4,ff=disable gpio function}
TX 取值{0=GPA0, 1=GPB7,2=GPC3, ff=disable gpio function}

描述：

BUART RX/TX GPIO 引脚复用设置

返回值：

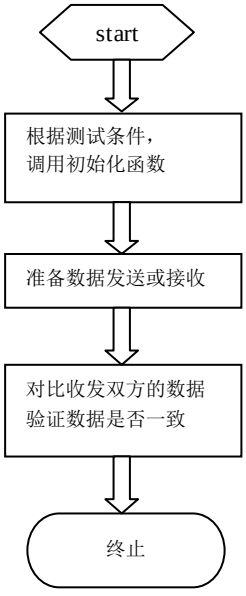
无

1.4 FUART 模块测试

1.4.1 测试 概述

Faurt 测试方法比较单一，就是根据不同的测试目标，不同的初始化函数以及部份的 I/O 控制函数进行控制.

1.4.2 测试 流程



1.5 中断处理函数示例

```
__attribute__((section(".driver.isr"))) //此行不能少，且不能变
VOID FuartInterrupt(VOID)           //名字要一致
{
    BYTE RxStat,TxStat;
    if(FuartIOctl(UART_IOCTL_RXSTAT_GET,0) & 1){
      /*
       * notify the task to take preparation to datum reception,and then task to
       *  receive the data indeed.
       */
      //OSQueueMsgSend(MSGDEV_FUART_DATRDY,NULL,0,MSGPRIO_LEVEL_L
O,0);
      //或者直接将数据先接收下来，Task 根据一些标志，再做处理，其中 Buffer 自定义
      if(0 < FuartRecvByte(&FuartRecvBuf[FuartRecvBufLen])){
        FuartRecvBufLen ++;
      }
      FuartIOctl(UART_IOCTL_RXINT_CLR,0);
    }
    if(FuartIOctl(UART_IOCTL_TXSTAT_GET,0) & 1){
      /*
       * notify the task that the data has transferred already
       */
        //TBD
    }
    //以下为错误处理
    RxStat = FuartIOctl(UART_IOCTL_RXSTAT_GET,0);
    if(RxStat & 0x1E){
      if(RxStat & 0x2){
        DBG(("Fuart RX error interrupt\r\n"));
      }
      else
      if(RxStat & 0x4){
        DBG(("Fuart RX error overrun\r\n"));
      }
      else
      if(RxStat & 0x8){
```

```
            DBG(("Fuart RX error Parity \r\n"));
          }
          else
          if(RxStat & 0x10){
            DBG(("Fuart RX error Frame \r\n"));
          }
        /*
         * clear FIFO before clear other flags
         */
        FuartIOctl(UART_IOCTL_RXFIFO_CLR,0);
        /*
         * clear other error flags
         */
        FuartIOctl(UART_IOCTL_RXINT_CLR,0);
    }
  }
```

1.6 FUART 波特率精度说明

不同的波特率需要设置不同的寄存器，这些操作已经被封装在了 lib 里面。不同的波特率下也有不同的理论误差。

系统时钟 fclk 为 96M，假设需要的波特率为 x，则 baud_clk 频率应该为 8x。baud_clk 的频率为 fclk 的分频时钟，分频系数为 (clk_div+1)+(clk_div_frac*0.125)=fclk/8x。假设所需波特率为 1382400，则 fclk/8x=8.68，则 clk_div 应该为 7，clk_div_frac 应该为 5。

典型波特率对应的参数配置以及允许的时钟频率偏差。其中，“+”表示比标准频率高(快)，“-”表示比标准频率低（慢）。

Baud rate	clk_div	clk_div_frac	Difference	Tolerance
1200	9999	0	0%	-2.5% ~ +3.4%
2400	4999	0	0%	-2.5% ~ +3.4%
4800	2499	0	0%	-2.5% ~ +3.4%
9600	1249	0	0%	-2.5% ~ +3.4%
19200	624	0	0%	-2.5% ~ +3.4%
38400	311	5	0%	-2.5% ~ +3.4%
57600	207	2	+0.05%	-2.45% ~ +2.85%
76800	155	2	0%	-2.5% ~ +3.4%
115200	103	1	+0.05%	-2.45% ~ +2.85%
230400	51	0	+0.17%	-2.33% ~ +3.57%
460800	25	0	0%	-2.5% ~ +3.4%
921600	12	0	0%	-2.5% ~ +3.4%
1382400	7	5	+0.7%	-1.8% ~ +3.5%
1843200	5	4	+0.17%	-2.33% ~ +2.97%
2764800	3	2	+2.13%	-0.37% ~ +4.93%
2764800	3	3	-0.8%	-3.3% ~ +2%