EC2X-QuecOpen

Uart 开发指导

LTE 系列

版本：EC2X-QuecOpen_uart_开发指导_V1.0

日期：2018-02-28

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市徐汇区虹梅路 1801 号宏业大厦 7 楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：
http://quectel.com/cn/support/sales.htm

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://quectel.com/cn/support/technical.htm
或发送邮件至：support@quectel.com

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

版权申明

文档历史

修订记录

版本	日期	作者	变更表述
1.0	2018-02-28	高飞虎	初始版本

目录

表格索引

图片索引

1 引言

文档从用户开发角度出发，介绍了硬件，软件驱动层，软件应用层等；可以帮助客户简易而快速的进行开发。

2 EC20 R2.1-QuecOpen 串口说明

EC20 R2.1-QuecOpen 模块提供 4 路串口：主串口、调试串口、串口 1 和串口 2，理论上最高可以支持到 4M 波特率。主串口、串口 1 和串口 2 功能相同，硬件上均支持 RTS/CTS，可作为外设通信串口。其中，串口 1 的 RTS/CTS 与 I2C 复用，串口 2 与 SPI 复用。如下描述了这四个串口的主要特性。

- 主串口支持 4800bps，9600bps，19200bps，38400bps，57600bps，115200bps，230400bps，460800bps 和 921600bps 波特率，默认波特率为 115200bps；支持 RTS，CTS 流控。用做普通外设通信串口。
- 调试串口支持 115200bps 波特率，用于 Linux 控制和日志输出。
- 串口 1 支持 4800bps，9600bps，19200bps，38400bps，57600bps，115200bps，230400bps，460800bps 和 921600bps 波特率，默认波特率为 115200bps；支持 RTS，CTS 流控。用做普通外设通信串口。
- 串口 2 支持 4800bps，9600bps，19200bps，38400bps，57600bps，115200bps，230400bps，460800bps 和 921600bps 波特率，默认波特率为 115200bps；支持 RTS，CTS 流控。用做普通外设通信串口。

3 硬件电路设计推荐

注意：QuecOpen 4G 模块 CTS 与 RTS 已经做了对调，所以用户在连的时候，是将 MCU 的 CTS 接到 4G 模块的 CTS 上，RTS 接 RTS;

3.1 带电平转换的串口设计电路

3.1.1 推荐使用 TI 公司的 TXS0108EPWR 电平转换芯片

EC20 R2.1-QuecOpen 模块的串口电平为 1.8V。若客户主机系统电平为 3.3V，则需在模块和主机的串口连接中增加电平转换器，推荐使用 TI 公司的 TXS0108EPWR。下图为使用电平转换芯片的参考电路设计。

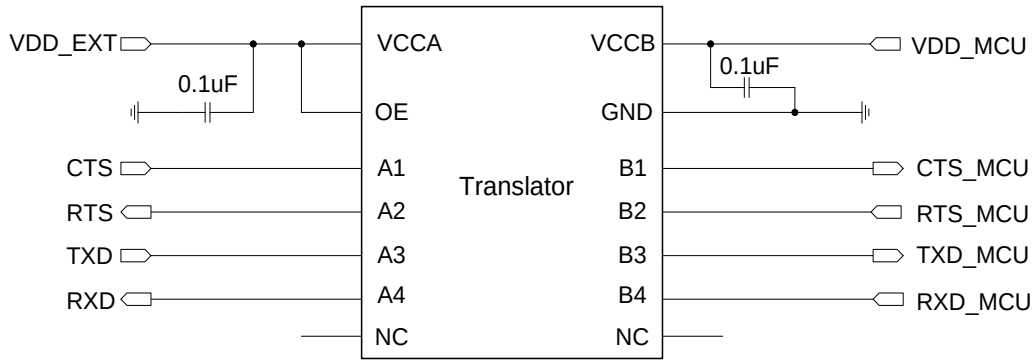


Figure 1: TXS0108EPWR 电平转换芯片电路

更多信息请访问 http://www.ti.com。

3.1.2 其他电平转换电路：三极管电平转换参考电路

另一种电平转换电路如下图所示。如下虚线部分的输入和输出电路设计可参考实线部分，但需注意连接方向。图中的三极管电平转换电路不适用于波特率超过 460Kbps 的应用。

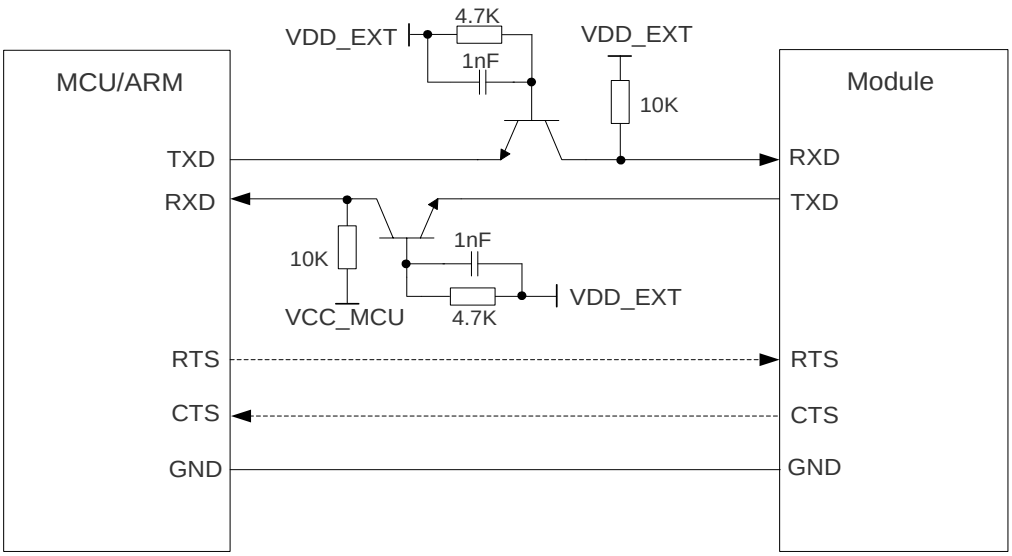


Figure 2：三极管电平转换参考电路

3.2 无电平转换芯片参考电路

模块的串口电平为 1.8V，若客户主机系统电平也为 1.8v，直连即可：

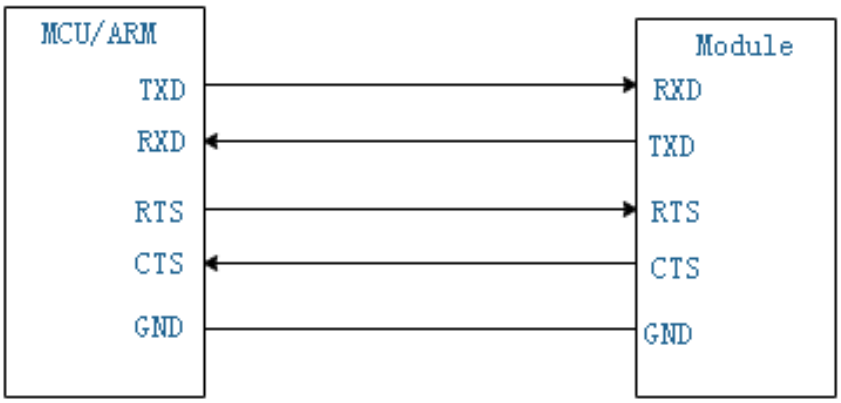


Figure 3: 1.8v 直连电路

4 驱动层及设备树软件适配

4.1 UART 管脚使用

4.1.1 重要说明

1.以下表格中，非默认的复用功能需要在软件配置后才有效，请参考对应的功能章节进行软件配置；
2.EC20 R2.1-QuecOpen 中，主串口不再具有 AT 功能，仅作为普通外设通信串口；
3.具体管脚使用参考《Quectel_EC20 R2.1_QuecOpen_GPIO_Assignment_Speadsheet》。

4.1.2 主串口引脚定义

Table 1 主串口引脚定义

管脚名称	管脚号	I/O	功能描述	
			复用功能 1（默认）	复用功能 2
MAIN_CTS	64	DO	UART_CTS_BLSP3	GPIO_3
MAIN_RTS	65	DI	UART_RTS_BLSP3	GPIO_2
MAIN_TXD	67	DO	UART_TXD_BLSP3	GPIO_0
MAIN_RXD	68	DI	UART_RXD_BLSP3	GPIO_1

4.1.3 调试串口引脚定义

Table 2 调试串口引脚定义

引脚名	引脚号	I/O	描述	备注
DBG_TXD	12	DO	模块发送数据	1.8V 电源域
DBG_RXD	11	DI	模块接收数据	1.8V 电源域

4.1.4 串口 1（RTS/CTS 与 I2C 复用）引脚定义

Table 3 串口 1 引脚定义

引脚名	引脚号	I/O	功能描述		
			复用功能 1（默认）	复用功能 2	复用功能 3
I2C_SCL	41	OD	I2C_SCL_BLSP2	GPIO_7	UART_CTS_BLSP2
I2C_SDA	42	OD	I2C_SDA_BLSP2	GPIO_6	UART_RTS_BLSP2
UART1_TXD	63	DO	GPIO_4	UART_TXD_BLSP2	UART_TXD_BLSP2
UART1_RXD	66	DI	GPIO_5	UART_RXD_BLSP2	UART_RXD_BLSP2

4.1.5 串口 2（与 SPI 复用）引脚定义

Table 4 串口 2 引脚定义

引脚名	引脚号	I/O	功能描述		
			复用功能 1（默认）	复用功能 2	复用功能 3
SPI_CS_N	37	DO	SPI_CS_N_BLSP6	GPIO_22	UART_RTS_BLSP6
SPI_MOSI	38	DO	SPI_MOSI_BLSP6	GPIO_20	UART_TXD_BLSP6
SPI_MISO	39	DI	SPI_MISO_BLSP6	GPIO_21	UART_RXD_BLSP6
SPI_CLK	40	DO	SPI_CLK_BLSP6	GPIO_23	UART_CTS_BLSP6

4.1.6 串口逻辑电平：

Table 5 串口逻辑电平

参数	最小值	最大值	单位
V_{IL}	-0.3	0.6	V
V_{IH}	1.2	2.0	V
V_{OL}	0	0.45	V
V_{OH}	1.35	1.8	V

4.2 UART 设备树配置方法

4.2.1 配置说明

关于串口的配置，如所兼容的 driver，引脚的选择，寄存器地址，串口中断号，CLK，以及系统休眠和工作时的配置等 quectel 都已经做好了，用户不需要关心太多，简单按以下方式关闭打开即可；

4.2.2 主串口配置

默认情况下主串口支持硬件流控，在模块内部显示为/dev/ttyHS0 设备节点。

```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /dev/tty32    /dev/tty43    /dev/tty54    /dev/tty8
/dev/tty11    /dev/tty22    /dev/tty33    /dev/tty44    /dev/tty55    /dev/tty9
/dev/tty12    /dev/tty23    /dev/tty34    /dev/tty45    /dev/tty56    /dev/ttyGS0
/dev/tty13    /dev/tty24    /dev/tty35    /dev/tty46    /dev/tty57    /dev/ttyHS0
/dev/tty14    /dev/tty25    /dev/tty36    /dev/tty47    /dev/tty58    /dev/ttyHSL0
/dev/tty15    /dev/tty26    /dev/tty37    /dev/tty48    /dev/tty59
/dev/tty16    /dev/tty27    /dev/tty38    /dev/tty49    /dev/tty6
/dev/tty17    /dev/tty28    /dev/tty39    /dev/tty5     /dev/tty60
/dev/tty18    /dev/tty29    /dev/tty4     /dev/tty50    /dev/tty61
```

一般情况下，我们推荐客户将此串口用到的 4 个管脚作为串口（默认情况）使用，或者修改为 GPIO 使用，虽然理论上是可以复用为 I2C 或者 SPI，但是我们不推荐客户这么做，我们提供了其他 I2C 和 SPI 硬件接口（请参考 i2c 或 spi 文档）；下面介绍一下如何修改主串口为 GPIO

```
--- a/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
+++ b/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
@@ -52,7 +52,7 @@
 };

 &blsp1_uart3 {
-       status = "ok";
+       status = "disabled";
 };
```

编译内核并烧录，具体参考《KBA_QuecOpen_EC2X&AG35_Quick_Start》编译方法；
烧录新固件，如下图发现/dev/ttyHS0 已经不存在，此时主串口已经关闭了，这 4 个管脚可以作为 GPIO 使用了；

```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /dev/tty32    /dev/tty43    /dev/tty54    /dev/tty8
/dev/tty11    /dev/tty22    /dev/tty33    /dev/tty44    /dev/tty55    /dev/tty9
/dev/tty12    /dev/tty23    /dev/tty34    /dev/tty45    /dev/tty56    /dev/ttyGS0
/dev/tty13    /dev/tty24    /dev/tty35    /dev/tty46    /dev/tty57    /dev/ttyHSL0
/dev/tty14    /dev/tty25    /dev/tty36    /dev/tty47    /dev/tty58
/dev/tty15    /dev/tty26    /dev/tty37    /dev/tty48    /dev/tty59
/dev/tty16    /dev/tty27    /dev/tty38    /dev/tty49    /dev/tty6
/dev/tty17    /dev/tty28    /dev/tty39    /dev/tty5     /dev/tty60
/dev/tty18    /dev/tty29    /dev/tty4     /dev/tty50    /dev/tty61
```

4.2.3 调试串口配置

默认用于模块 Linux 调试和日志输出，不建议做任何改动；
调试串口未使能硬件流控，在模块内部显示为/dev/ttyHSL0 设备节点；

```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /dev/tty32    /dev/tty43    /dev/tty54    /dev/tty8
/dev/tty11    /dev/tty22    /dev/tty33    /dev/tty44    /dev/tty55    /dev/tty9
/dev/tty12    /dev/tty23    /dev/tty34    /dev/tty45    /dev/tty56    /dev/ttyGS0
/dev/tty13    /dev/tty24    /dev/tty35    /dev/tty46    /dev/tty57    /dev/ttyHS0
/dev/tty14    /dev/tty25    /dev/tty36    /dev/tty47    /dev/tty58    /dev/ttyHSL0
/dev/tty15    /dev/tty26    /dev/tty37    /dev/tty48    /dev/tty59
/dev/tty16    /dev/tty27    /dev/tty38    /dev/tty49    /dev/tty6
/dev/tty17    /dev/tty28    /dev/tty39    /dev/tty5     /dev/tty60
/dev/tty18    /dev/tty29    /dev/tty4     /dev/tty50    /dev/tty61
```

此串口在设备树对应的是：
文件路径：ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi

```
32 &blsp1_uart5 {
33         status = "ok";
34         pinctrl-names ="sleep", "default";
35         pinctrl-0 = <&uart_console_sleep>;
36         pinctrl-1 = <&uart_console_active>;
37 };
```

4.2.4 串口 1 配置

默认情况下串口 1 未使能，在模块内部不显示设备节点，PIN63,PIN66 为 GPIO function；
且此串口的 CTS/RTS 与 I2C 复用，所以使用 I2C 的情况下，此串口不支持硬件流控；

```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /dev/tty32    /dev/tty43    /dev/tty54    /dev/tty8
/dev/tty11    /dev/tty22    /dev/tty33    /dev/tty44    /dev/tty55    /dev/tty9
/dev/tty12    /dev/tty23    /dev/tty34    /dev/tty45    /dev/tty56    /dev/ttyGS0
/dev/tty13    /dev/tty24    /dev/tty35    /dev/tty46    /dev/tty57    /dev/ttyHS0
/dev/tty14    /dev/tty25    /dev/tty36    /dev/tty47    /dev/tty58    /dev/ttyHSL0
/dev/tty15    /dev/tty26    /dev/tty37    /dev/tty48    /dev/tty59
/dev/tty16    /dev/tty27    /dev/tty38    /dev/tty49    /dev/tty6
/dev/tty17    /dev/tty28    /dev/tty39    /dev/tty5     /dev/tty60
/dev/tty18    /dev/tty29    /dev/tty4     /dev/tty50    /dev/tty61
```

若用户需要增加额外串口按照以下方式可以打开此串口：

```
--- a/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
+++ b/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
@@ -56,7 +56,7 @@
 };

 &blsp1_uart2 {
-        status = "disabled";    //if need, user can enable by themselves
+        status = "ok";  //if need, user can enable by themselves
         pinctrl-names ="sleep", "default";
         pinctrl-0 = <&blsp1_uart2_sleep>;
         pinctrl-1 = <&blsp1_uart2_active>;
```

编译内核并烧录，具体参考《KBA_QuecOpen_EC2X&AG35_Quick_Start》编译方法；
烧录新固件，如下图发现多出/dev/ttyHSL1 设备节点，此时串口 1 可以工作了；

```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /dev/tty32    /dev/tty43    /dev/tty54    /dev/tty8
/dev/tty11    /dev/tty22    /dev/tty33    /dev/tty44    /dev/tty55    /dev/tty9
/dev/tty12    /dev/tty23    /dev/tty34    /dev/tty45    /dev/tty56    /dev/ttyGS0
/dev/tty13    /dev/tty24    /dev/tty35    /dev/tty46    /dev/tty57    /dev/ttyHS0
/dev/tty14    /dev/tty25    /dev/tty36    /dev/tty47    /dev/tty58    /dev/ttyHSL0
/dev/tty15    /dev/tty26    /dev/tty37    /dev/tty48    /dev/tty59    /dev/ttyHSL1
/dev/tty16    /dev/tty27    /dev/tty38    /dev/tty49    /dev/tty6
/dev/tty17    /dev/tty28    /dev/tty39    /dev/tty5     /dev/tty60
/dev/tty18    /dev/tty29    /dev/tty4     /dev/tty50    /dev/tty61
```

4.2.5 串口 2 配置

默认情况下串口 2 未使能，在模块内部不显示串口设备节点，PIN37,PIN38,PIN39,PIN40 被配置为 SPI 接口；

```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /dev/tty32    /dev/tty43    /dev/tty54    /dev/tty8
/dev/tty11    /dev/tty22    /dev/tty33    /dev/tty44    /dev/tty55    /dev/tty9
/dev/tty12    /dev/tty23    /dev/tty34    /dev/tty45    /dev/tty56    /dev/ttyGS0
/dev/tty13    /dev/tty24    /dev/tty35    /dev/tty46    /dev/tty57    /dev/ttyHS0
/dev/tty14    /dev/tty25    /dev/tty36    /dev/tty47    /dev/tty58    /dev/ttyHSL0
/dev/tty15    /dev/tty26    /dev/tty37    /dev/tty48    /dev/tty59
/dev/tty16    /dev/tty27    /dev/tty38    /dev/tty49    /dev/tty6
/dev/tty17    /dev/tty28    /dev/tty39    /dev/tty5     /dev/tty60
/dev/tty18    /dev/tty29    /dev/tty4     /dev/tty50    /dev/tty61
```

若用户需要增加额外串口按照以下方式可以打开此串口：

```
--- a/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
+++ b/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
@@ -48,7 +48,7 @@
 //2016-01-19, comment out by jun.wu, remove UART3 && spi_1 from device tree

 &spi_6 {
-        status = "ok";
+        status = "disabled";
 };

 &blsp1_uart3 {
@@ -63,7 +63,7 @@
 };

 &blsp1_uart6 {
-        //status = "ok";
+        status = "ok";
         pinctrl-names ="default", "sleep";
         pinctrl-0 = <&blsp1_uart6_active>;
         pinctrl-1 = <&blsp1_uart6_sleep>;
```

编译内核并烧录，具体参考《KBA_QuecOpen_EC2X&AG35_Quick_Start》编译方法；
烧录新固件，如下图发现多出/dev/ttyHSL2 设备节点，此时串口 2 可以工作了；
另外需要说明的是，模块里面看到的串口序号是串口 driver 根据设备树里多个串口节点排列顺序决定的，若上面的串口 1 未使能，那么串口 2 节点名称将变为 ttyHSL1；

```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /dev/tty32    /dev/tty43    /dev/tty54    /dev/tty8
/dev/tty11    /dev/tty22    /dev/tty33    /dev/tty44    /dev/tty55    /dev/tty9
/dev/tty12    /dev/tty23    /dev/tty34    /dev/tty45    /dev/tty56    /dev/ttyGS0
/dev/tty13    /dev/tty24    /dev/tty35    /dev/tty46    /dev/tty57    /dev/ttyHS0
/dev/tty14    /dev/tty25    /dev/tty36    /dev/tty47    /dev/tty58    /dev/ttyHSL0
/dev/tty15    /dev/tty26    /dev/tty37    /dev/tty48    /dev/tty59    /dev/ttyHSL1
/dev/tty16    /dev/tty27    /dev/tty38    /dev/tty49    /dev/tty6     /dev/ttyHSL2
/dev/tty17    /dev/tty28    /dev/tty39    /dev/tty5     /dev/tty60
/dev/tty18    /dev/tty29    /dev/tty4     /dev/tty50    /dev/tty61
```

5 QuecOpen 应用层 API

5.1 用户编程说明

QuecOpen 项目 SDK 中提供了一套完整的用户编程接口；
参考路径：ql-ol-sdk/ql-ol-extsdk/

```
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk$ ls
docs  example  include  lib  target  tools
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk$
```

图中的 lib 目录下包含 quectel 提供的 API 接口库；include 目录是所有 API 的头文件；example 目录是提供的按功能划分的 API 使用示例；

这里我只介绍关于 UART 相关的接口以及参考示例；
在进行 uart 应用程序的编写需要依赖库 libql_peripheral.a；
头文件：ql_uart.h

5.2 UART API 介绍

流控枚举：选择流控模式

```
typedef enum {
    FC_NONE = 0,    // None Flow Control
    FC_RTSCTS,      // Hardware Flow Control (rtscts)
    FC_XONXOFF      // Software Flow Control (xon/xoff)
}Enum_FlowCtrl;
```

检验位枚举：支持 none,奇校验和偶校验

```
typedef enum {
    PB_NONE = 0,    //none parity check
    PB_ODD,         //odd parity check
    PB_EVEN         //even parity check
}Enum_ParityBit;
```

数据位枚举：支持 5,6,7,8

```
typedef enum {
    DB_CS5 = 5,
    DB_CS6 = 6,
    DB_CS7 = 7,
    DB_CS8 = 8
}Enum_DataBit;
```

停止位枚举：支持 1bit,2bit

```
typedef enum {
    SB_1 = 1,
    SB_2 = 2
}Enum_StopBit;
```

波特率枚举：模块支持的波特率

```
typedef enum {
    B_300       = 300,
    B_600       = 600,
    B_1200      = 1200,
    B_2400      = 2400,
    B_4800      = 4800,
    B_9600      = 9600,
    B_19200     = 19200,
    B_38400     = 38400,
    B_57600     = 57600,
    B_115200    = 115200,
    B_230400    = 230400,
    B_460800    = 460800,
    B_921600    = 921600,
    B_3000000   = 3000000,
    B_4000000   = 4000000,
}Enum_BaudRat;
```

串口属性结构体：

```
typedef struct {
    Enum_BaudRate       baudrate;
    Enum_DataBit        databit;
    Enum_StopBit        stopbit;
    Enum_ParityBit      parity;
    Enum_FlowCtrl       flowctrl;
}ST_UARTDCB;
```

int Ql_UART_Open(const char* port, Enum_BaudRate baudrate, Enum_FlowCtrl flowctrl);
以指定波特率及流控方式打开某串口设备文件，默认检验位为 PB_NONE, 数据位 DB_CS8，停止位 SB_1,
若需要修改这几个属性，使用 int Ql_UART_SetDCB(int fd, ST_UARTDCB *dcb)接口；
参数： port: 设备名，如：/dev/ttyHS0
baudrate: 波特率，见枚举 Enum_BaudRat，如：B_9600, B_115200
flowCtrl: 流控，枚举 Enum_FlowCtrl
返回值：返回文件描述符，否则返回-1；

int Ql_UART_SetDCB(int fd, ST_UARTDCB *dcb)；
设置串口属性，包括流控，校验位，数据位，停止位，波特率；
参数： fd: 设备文件描述符；

dcb: 填充的串口属性结构体；
返回值：返回 0，出错返回-1；

int Ql_UART_GetDCB(int fd, ST_UARTDCB *dcb)；
获取当前串口属性，包括流控，校验位，数据位，停止位，波特率；
参数： fd: 设备文件描述符；
dcb: 串口属性结构体；
返回值：返回 0，出错返回-1；

int Ql_UART_Read(int fd, char* buf, unsigned int buf_len);
从串口读取指定字节长度内容到 buf；返回读取的字节长度；
参数： fd: 设备文件描述符；
buf: 读数据指针；
buf_len: 读取长度；
返回值：返回读取的字节长度；

int Ql_UART_Write(int fd, const char* buf, unsigned int buf_len);
从 buf 中写指定长度数据到串口；返回写入的字节长度；
参数： fd: 设备文件描述符；
buf: 写数据指针；
buf_len: 写入长度；
返回值：返回写入的字节长度；

int Ql_UART_Close(int fd);
关闭设备文件描述符。

高级串口编程：
以上接口完全可以满足客户的需求；
若果客户为高级 Linux 用户且对串口的特性十分精通，那么客户可以自行使用下面的接口去设置串口；
int Ql_UART_IoCtl(int fd, unsigned int cmd, void* pValue);
控制串口设备属性；
参数： fd: 设备文件描述符；
cmd: 串口 ioctl 请求，如：TCGETS，TCSETS，TIOCMGET，TIOCMSET 等；
pValue: 串口设备属性指针
返回值：成功返回 0，错误返回-1；

用户可以监听 uart_fd 设备文件描述符，来实现异步通知读取串口数据
参考：ql-ol-extsdk/example/uart

6 UART 功能测试验证

6.1 example 介绍及编译

Example 以主串口为例即/dev/ttyHS0
示例中主线程每隔一秒向串口写数据，打开上位机 COM 口可以收到数据；同时子线程在监听 rx 是否有数据进来，在上位机 COM 口发送数据进来，主串口会收到并打印出来；

```
3
4 #define QL_UART1_DEV  "/dev/ttyHS0"
5
6 static int fd_uart = -1;
```

进入到 ql-ol-sdk/ql-ol-extsdk/example/uart 目录，make 生成 example_uart 可执行程序，可以编译的前提必须是之前进行了交叉编译环境的初始化
source ql-ol-crosstool/ql-ol-crosstool-env-init

```
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk/example/uart$ make
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon  -O2 -fexpensive-
de -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-o
eon-oe-linux-gnueabi/usr/include -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosst
-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/usr/include/dsutils -I/home/gale
home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linu
-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/usr/include -I/home/gale/MDM9x07
M9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/us
gnueabi/usr/include/dsutils -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/s
osstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/usr/include/qmi-framework  -L./ -L/hom
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon -L./ -L/home/gale
9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk/example/uart/../../lib -lrt -lpthread /home/gale
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk/example/uart$
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk/example/uart$
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk/example/uart$ ls
example_uart  example_uart.c  example_uart.o  Makefile
```

6.2 功能测试

6.2.1 未使能流控

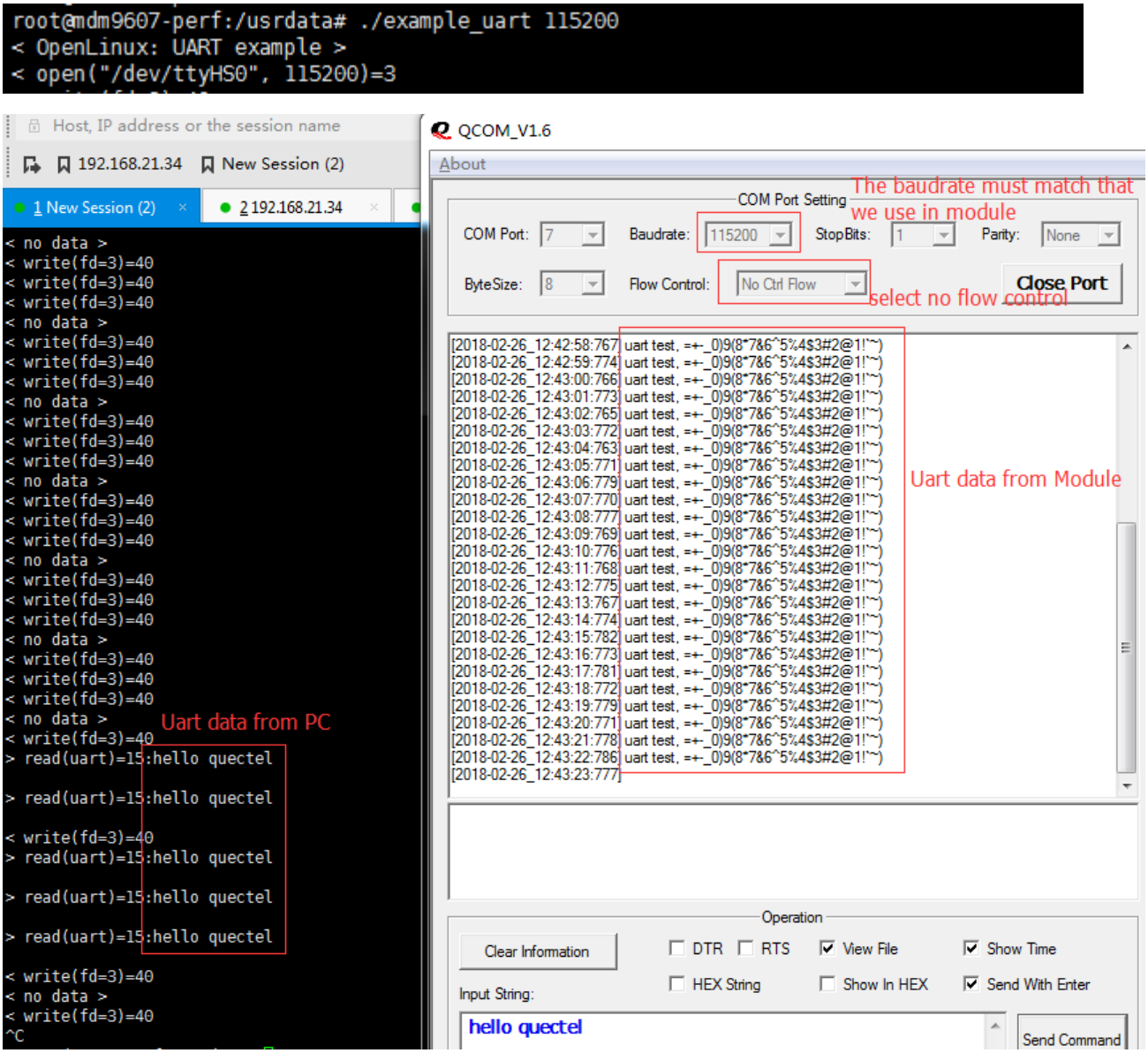
```
        fd_uart = Ql_UART_Open(QL_UART1_DEV, baudRate, FC_NONE);
        printf("< open(\"%s\", %d)=%d\n", QL_UART1_DEV, baudRate, fd_uart);
```

1. 编译上传 example_uart 到模块
   使用 adb push <example_uart 在上位机路径> <模块内部路径，如/usrdata>
   或者
   使用串口协议 rz 上传

2. 若使用 OPEN_EVB，需要用跳线帽连接 J0202 的 main uart 排针

GPIO_0 连接 MAIN_TXD
GPIO_1 连接 MAIN_RXD
来连通硬件通路

3. 执行 example_uart 115200，并在上位机使用对应的波特率以及无流控方式打开对应 COM 口，如下图



6.2.2 使能硬件流程

修改 example 来使能硬件流控；

```
fd_uart = Ql_UART_Open(QL_UART1_DEV, baudRate, FC_RTSCTS);
printf("< open(\"%s\", %d)=%d\n", QL_UART1_DEV, baudRate, fd_uart);
```

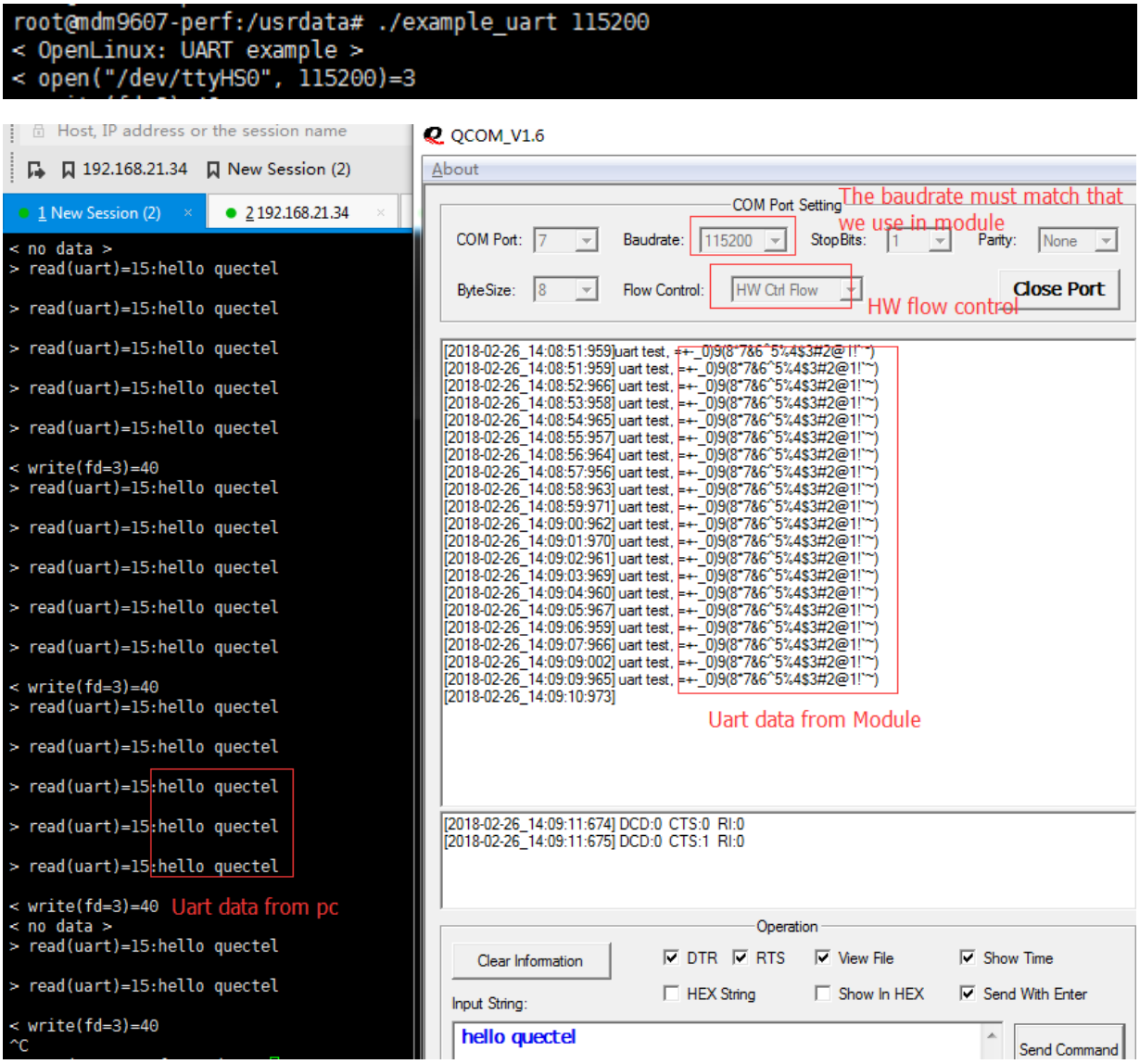
1. 编译上传 example_uart 到模块
使用 adb push <example_uart 在上位机路径> <模块内部路径，如/usrdata>
或者

使用串口协议 rz 上传

2. 若使用 OPEN_EVB，需要用跳线帽连接 J0202 的 main uart 排针

GPIO_0 连接 MAIN_TXD
GPIO_1 连接 MAIN_RXD
GPIO_2 连接 MAIN_RTS
GPIO_3 连接 MAIN_CTS
来连通硬件通路

3. 执行 example_uart 115200，并在上位机使用对应的波特率以及硬件流控方式打开对应 COM 口，如下图



6.2.3 使能软件流控

6.2.3.1 软件流控 XON/XOFF 字符说明

XOFF/XON representations in ASCII

Code	Meaning	ASCII	Dec	Hex	Keyboard
XOFF	Pause transmission	DC3	19	13	Ctrl + S
XON	Resume transmission	DC1	17	11	Ctrl + Q

6.2.3.2 软件流控测试

修改 example 来使能软件流控；

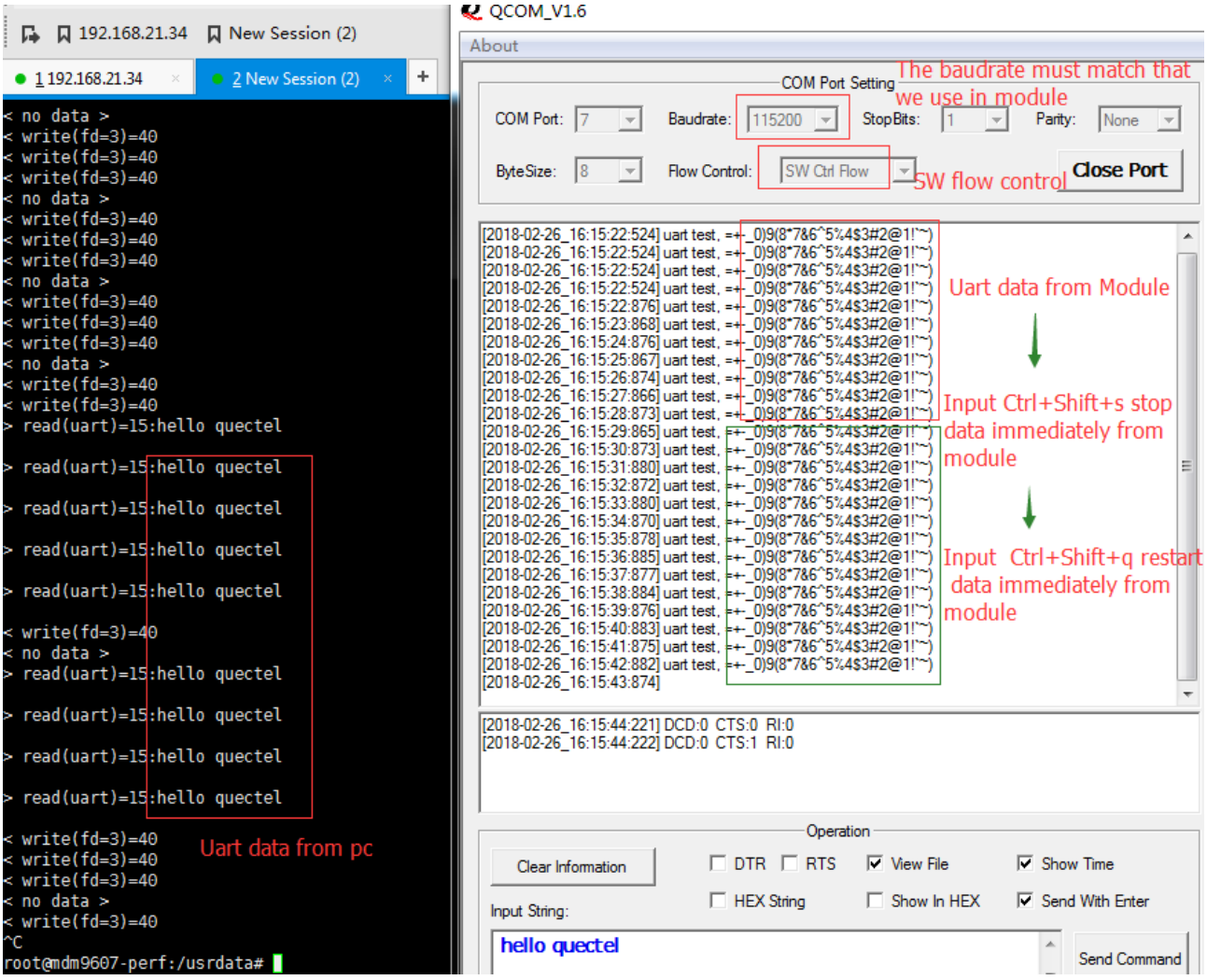
```
fd_uart = Ql_UART_Open(QL_UART1_DEV, baudRate, FC_XONXOFF);
printf("< open(\"%s\", %d)=%d\n", QL_UART1_DEV, baudRate, fd_uart);
```

1. 编译上传 example_uart 到模块
   使用 adb push <example_uart 在上位机路径> <模块内部路径，如/usrdata>
   或者
   使用串口协议 rz 上传

2. 若使用 OPEN_EVB，需要用跳线帽连接 J0202 的 main uart 排针
   GPIO_0 连接 MAIN_TXD
   GPIO_1 连接 MAIN_RXD
   来连通硬件通路
3. 执行 example_uart 115200，并在上位机使用对应的波特率以及硬件流控方式打开对应 COM 口，如下图

```
root@mdm9607-perf:/usrdata# ./example_uart 115200
< OpenLinux: UART example >
< open("/dev/ttyHS0", 115200)=3
```

开启软件流控传输数据时，在上位机串口软件中键盘键入 Ctrl+Shift+S ，模块端将立刻停止数据发送；Ctrl+Shift+Q 又会恢复数据传输，验证软件流控正常；


192.168.21.34
New Session (2)
1 192.168.21.34
2 New Session (2)
< no data >
< write(fd=3)=40
< write(fd=3)=40
< write(fd=3)=40
< no data >
< write(fd=3)=40
< write(fd=3)=40
< write(fd=3)=40
< no data >
< write(fd=3)=40
< write(fd=3)=40
< write(fd=3)=40
< no data >
< write(fd=3)=40
< write(fd=3)=40
> read(uart)=15:hello quectel
> read(uart)=15:hello quectel
> read(uart)=15:hello quectel
> read(uart)=15:hello quectel
> read(uart)=15:hello quectel
< write(fd=3)=40
< no data >
> read(uart)=15:hello quectel
> read(uart)=15:hello quectel
> read(uart)=15:hello quectel
> read(uart)=15:hello quectel
< write(fd=3)=40
< write(fd=3)=40
< write(fd=3)=40
< no data >
< write(fd=3)=40
^C
root@mdm9607-perf:/usrdata#
Uart data from pc
QCOM_V1.6
About
COM Port Setting
The baudrate must match that we use in module
COM Port: 7
Baudrate: 115200
StopBits: 1
Parity: None
ByteSize: 8
Flow Control: SW Ctrl Flow
SW flow control
Close Port
[2018-02-26_16:15:22:524] uart test, =+-_0)9(8*7&6^5%4$3#2@1!~)
[2018-02-26_16:15:43:874]
Uart data from Module
Input Ctrl+Shift+s stop data immediately from module
Input Ctrl+Shift+q restart data immediately from module
[2018-02-26_16:15:44:221] DCD:0 CTS:0 RI:0
[2018-02-26_16:15:44:222] DCD:0 CTS:1 RI:0
Operation
Clear Infomation
DTR
RTS
View File
Show Time
HEX String
Show In HEX
Send With Enter
Input String:
hello quectel
Send Command