EC2X-QuecOpen UART Development Guide

LTE Standard Module Series

Rev. EC2x-QuecOpen_UART_ Development_Guide_V1.0

Date: 2018-02-28

Status: Preliminary

www.quectel.com

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to: support@quectel.com

Quectel Confidential

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

THE INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF QUECTEL WIRELESS SOLUTIONS CO., LTD. TRANSMITTING, REPRODUCTION, DISSEMINATION AND EDITING OF THIS DOCUMENT AS WELL AS UTILIZATION OF THE CONTENT ARE FORBIDDEN WITHOUT PERMISSION. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

Copyright © Quectel Wireless Solutions Co., Ltd. 2019. All rights reserved.

About the Document

History

Revision	Date	Author	Description
1.0	2018-02-28	Gale GAO	Initial

Quectel
Confidential

Contents

Quectel Confidential

Table Index

Quectel
Confidential

Figure Index

Quectel
Confidential

1 Introduction

This document mainly introduces hardware, software driver layer, software application layer from the perspective of users to help them develop easily and quickly.

This document mainly applies for global market. The LTE Standard module currently supporting this interface includes:

- EC2x: EC10 R2.1/EC25/EC21

Quectel Confidential

2 EC20 R2.1-QuecOpen UART Description

Four UART that are provided by EC20 R2.1-QuecOpen module are: main UART, debug UART, UART1, and UART2. Theoretically, the maximum baud rate supported is 4M. The main UART has the same function with UART1 and UART2, and the three UART that can be used as peripheral communication UART support RTS/CTS on hardware. Among them, the RTS/CTS of UART1 is multiplexed with I2C, UART2 multiplexed with SPI. The main features of the four UART are described as follows:

- The baud rates supported by main UART are: 4800bps, 9600bps, 19200bps, 38400bps, 57600bps, 115200bps, 230400bps, 460800bps and 921600bps. The default baud rate is 115200bps. Main UART that can be used as common peripheral communication UART supports RTS/CTS flow control.
- The baud rate supported by debug UART that is used for Linux control and log output is 115200bps.
- The baud rates supported by UART1 are: 4800bps, 9600bps, 19200bps, 38400bps, 57600bps, 115200bps, 230400bps, 460800bps and 921600bps. The default baud rate is 115200bps. UART1 that can be used as common peripheral communication UART supports RTS/CTS flow control.
- The baud rates supported by UART2 are: 4800bps, 9600bps, 19200bps, 38400bps, 57600bps, 115200bps, 230400bps, 460800bps and 921600bps. The default baud rate is 115200bps. UART2 that can be used as common peripheral communication UART supports RTS/CTS flow control.

Quectel
Confidential

3 Recommendations for Hardware Circuit Design

Note: CTS and RTS on 4G module of QuecOpen have been reversed, so user should connect the CTS of the MCU to the CTS on 4G module, the RTS of the MCU to the RTS on 4G module when connecting.

3.1. UART Design Circuit with Level Shifting

3.1.1. TXS0108EPWR Level Shifting Chip made by TI Company Being Recommended

The UART level of EC20 R2.1-QuecOpen module is 1.8V. If the user's host system level is 3.3V, it is necessary to add Level shifter when connecting module to host UART. It is recommended to use the TXS0108EPWR Level Shifting Chip made by TI Company. The following figure shows the reference circuit design with level shifting chip.

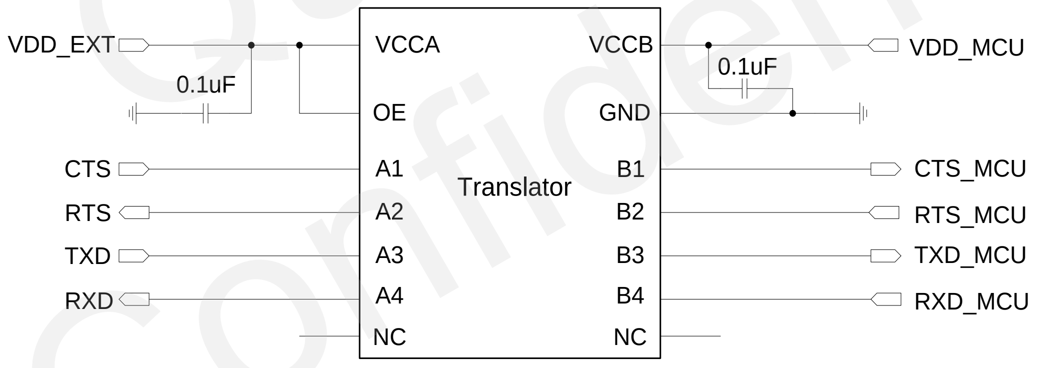


Figure 1: TXS0108EPWR Level Shifting Chip Circuit

For more details, please visit http://www.ti.com.

3.1.2. Other Level Shifting Circuits: Reference Circuit of Triode Level Shifting

Another level shifting circuit is shown in the following figure. The input and output circuit design in the following dotted line can refer to the part of solid line, but please pay attention to the connection direction. The triode level shifting circuit in the following figure is not applied to applications with a baud rate exceeding 460Kbps.

Quectel Confidential

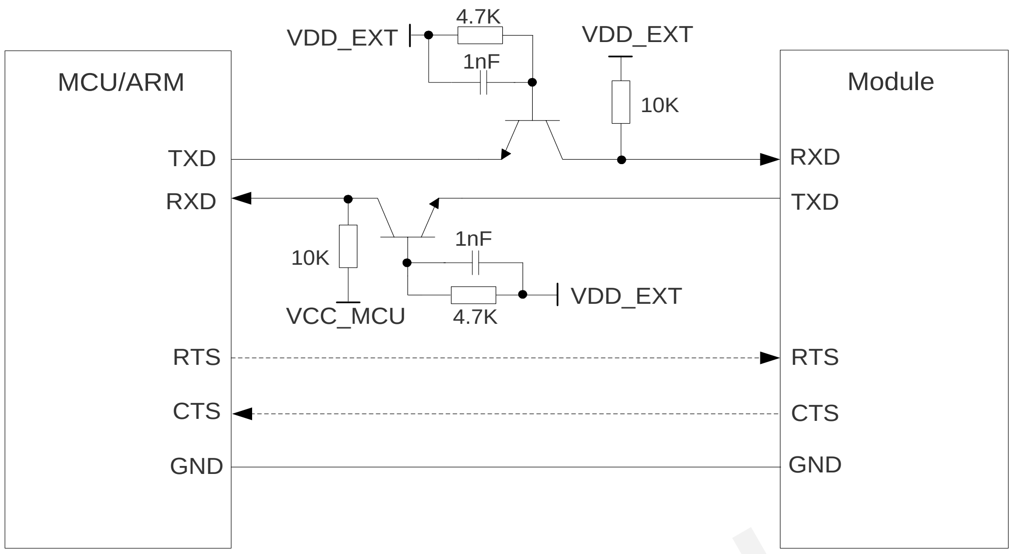


Figure 2: Triode Level Shifting Reference Circuit

3.2. Reference Circuit of Non-Level Shifting Chip

Module's UART level is 1.8V, if the user's host system level is also 1.8V, please connect directly:

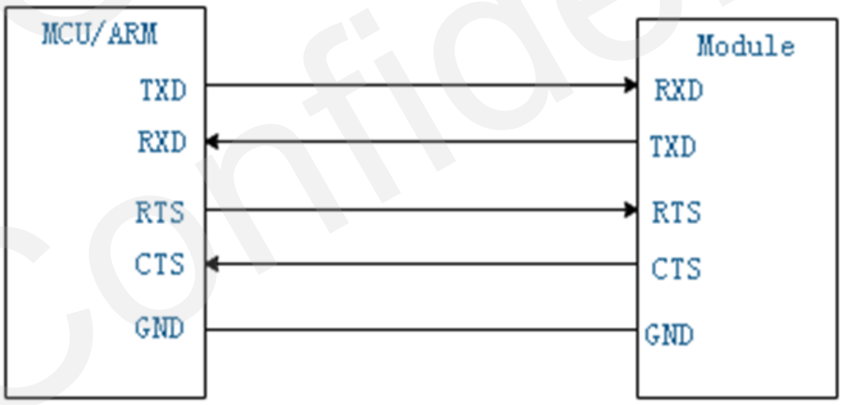


Figure 3: 1.8V Direct Connection Circuit

Quectel Confidential

4 Software Adaptation for Driver Layer and Device Tree

4.1. UART Pin Use

4.1.1. NOTICES

- Non-default multiplexing functions are valid with the condition of software being configured in the following table. For software configuration, please refer to the corresponding function chapter.
- In EC20 R2.1-QuecOpen, the main UART is only used as common peripheral communication UART and no longer has the AT function.
- For the specific pin usage, please refer to *Quectel_EC20 R2.1_QuecOpen_GPIO_Assignment_ Speadsheet*.

4.1.2. Pin Definition of Main UART

Table 1: Pin Definition of Main UART

Pin Name	Pin Number	I/O	Functional Description	
			Multiplex Function 1 (Default)	Multiplex Function 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. Pin Definition of Debug UART

Table 2: Pin Definition of Debug UART

Pin Name	Pin Number	I/O	Description	Remarks

Quectel Confidential

DBG_TXD	12	DO	Modules send data	1.8V Power Domain
DBG_RXD	11	DI	Modules send data	1.8V Power Domain

4.1.4. Pin Definition of UART1 (RTS/CTS Multiplexing with I2C)

Table 3: Pin Definition of UART1

Pin Name	Pin Number	I/O	Functional Description		
			Multiplex Function 1 (Default)	Multiplex Function 2	Multiplex Function 2
I2C_SCL	41	OD	I2C_SCL_BLSP2	GPIO_7	UART_CTS_BLSP 2
I2C_SDA	42	OD	I2C_SDA_BLSP2	GPIO_6	UART_RTS_BLSP 2
UART1_TXD	63	DO	GPIO_4	UART_TXD_BLS P2	UART_TXD_BLSP 2
UART1_RXD	66	DI	GPIO_5	UART_RXD_BL SP2	UART_RXD_BLSP 2

4.1.5. Pin Definition of UART2 (Multiplexing with SPI)

Table 4: Pin Definition of UART2

Pin Name	Pin Number	I/O	Functional Description		
			Multiplex Function 1 (Default)	Multiplex Function 2	Multiplex Function 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

Quectel Confidential

4.1.6. UART Logic Level

Table 5: UART Logic Level

Parameter	Minimum	Maximum	Unit
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

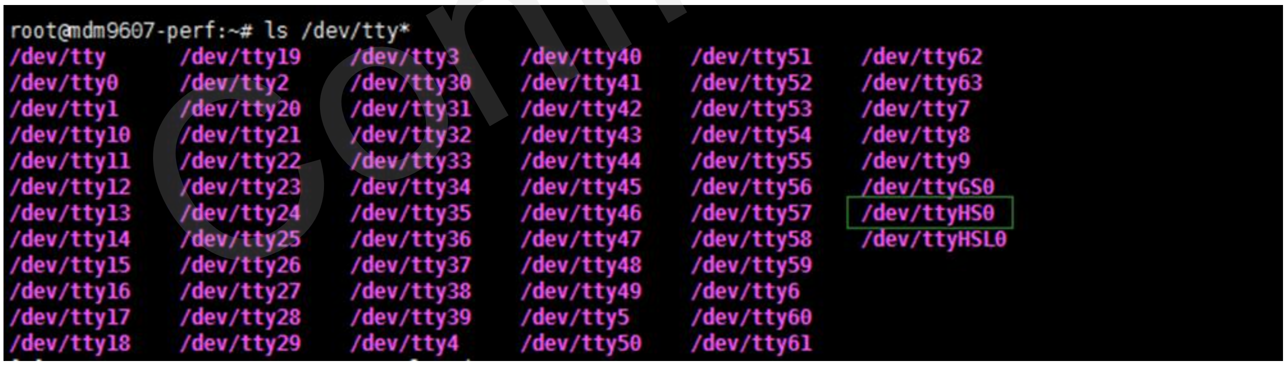
4.2. UART Device Tree Configuration Method

4.2.1. Configuration Instructions

Quectel has completed a series of UART configuration such as its compatible drivers, pin selection, register address, UART interrupt number, CLK, and system sleeping and working configuration, etc., so users just need to simply open or close as follows.

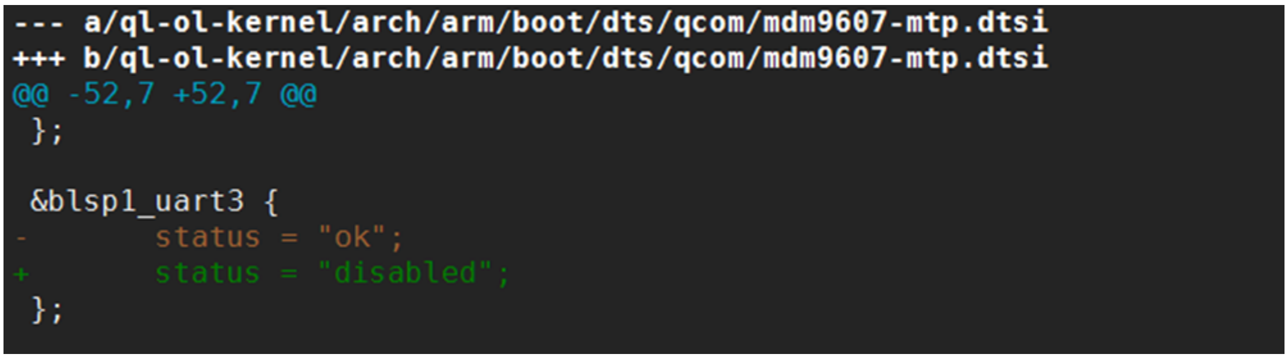
4.2.2. Main UART Configuration

Main UART supports hardware flow control by default and is shown as the device node of */dev/ttyHS0* inside the module.



```
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
```

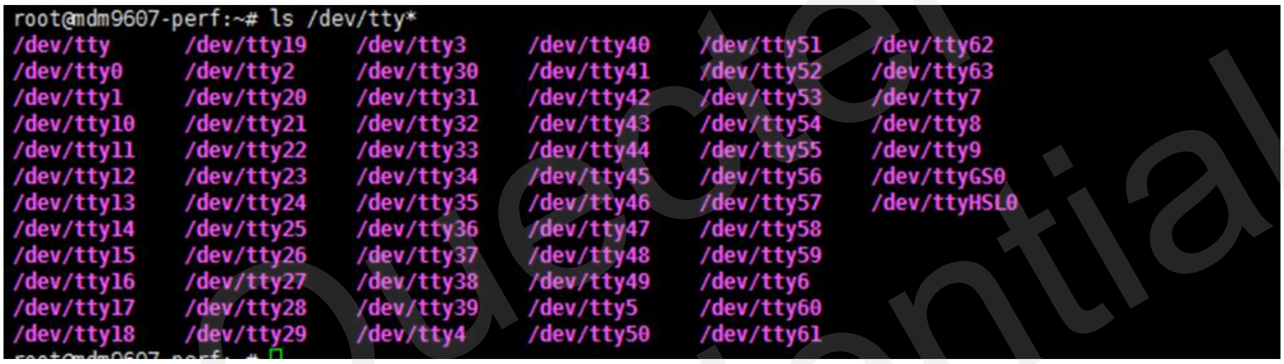
Generally Speaking, it is recommended to treat the 4 pins that is used the UART as UART (by default) or modify the 4 pins as GPIO. It is not recommended to multiplex to I2C or SPI, and Quectel provides other I2C and SPI hardware interfaces (please refer to the i2c or spi documentation). Here introduces how to modify main UART as 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";
 };
```

Compile the kernel and download it. For more details, please refer to the compilation method in *KBA_QuecOpen_EC2X&AG35_Quick_Start*.

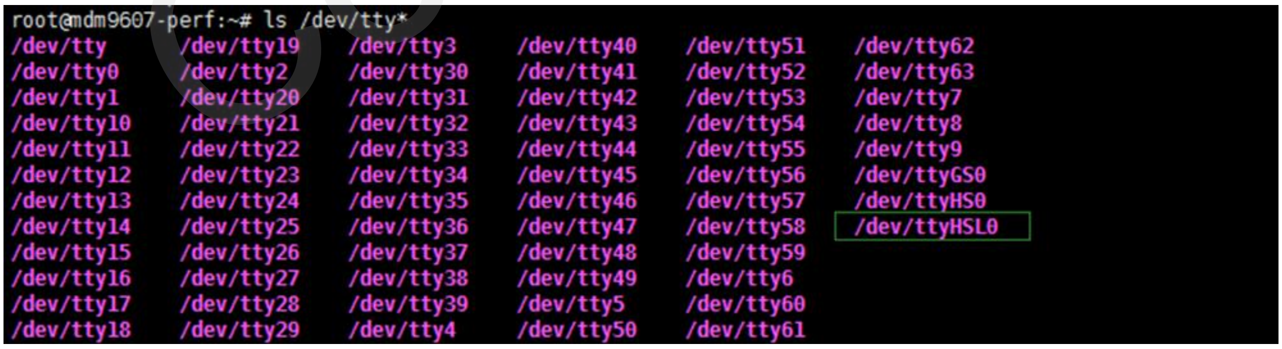
Download new firmware, and if /dev/ttyHS0 disappeared as the following figure, which means main UART has been closed and the 4 pins can be used as GPIO.



```
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. Debug UART Configuration

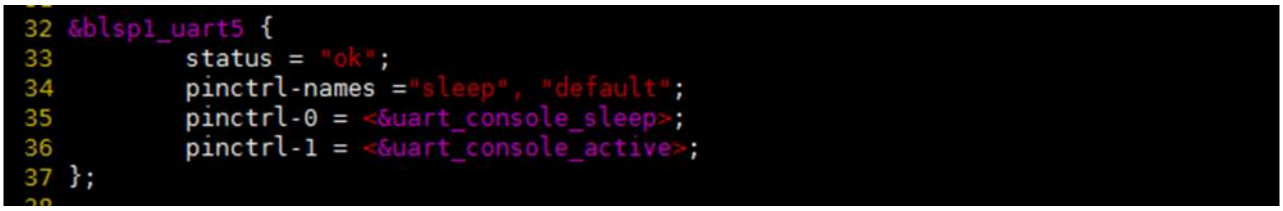
Debug UART is used for Linux debugging and log output of the module by default, it is not recommended to make any changes.
Debug UART does not enable hardware flow control and is shown as the device node of */dev/ttyHSL0* inside the module.



```
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
```

The corresponding UART in device tree is:
File path: *ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi*

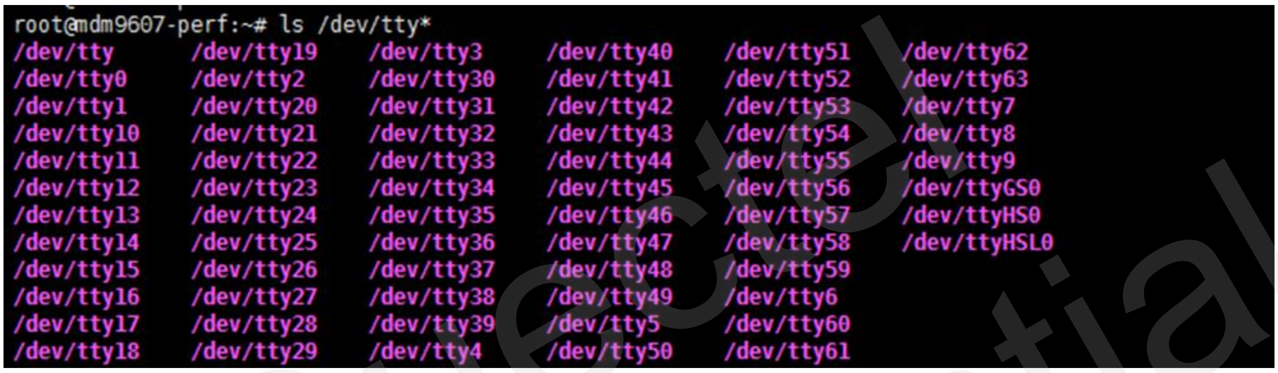
Quectel Confidential



```
&blsp1_uart5 {
        status = "ok";
        pinctrl-names ="sleep", "default";
        pinctrl-0 = <&uart_console_sleep>;
        pinctrl-1 = <&uart_console_active>;
};
```

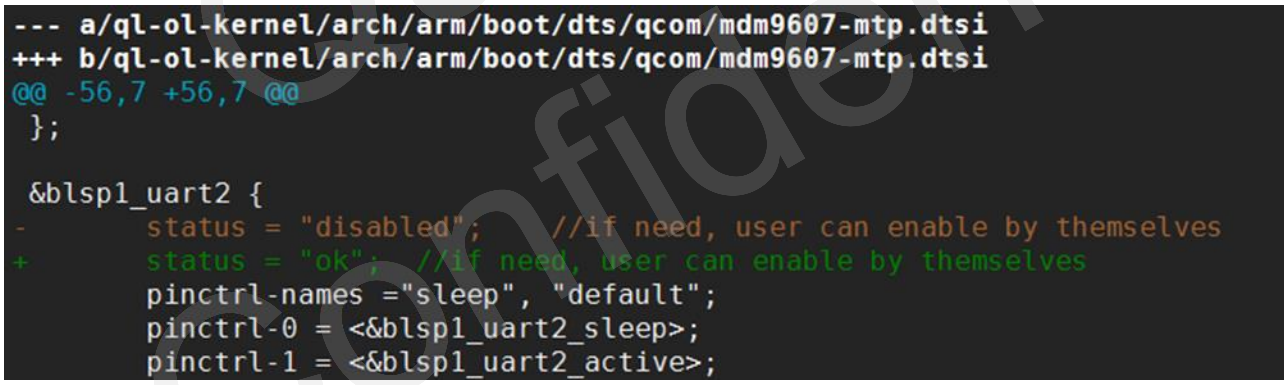
4.2.4. UART1 Configuration

UART1 is not enabled by default and UART device node are not displayed inside the module. PIN63 and PIN66 are GPIO function.
CTS/RTS of the UART is multiplexed with I2C, so the UART does not support hardware flow control in the case of using I2C.



```
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
```

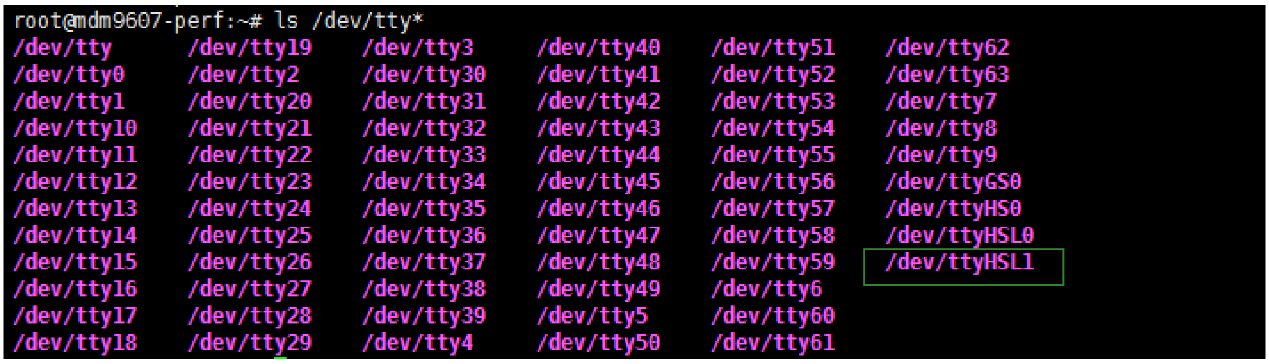
If users need to add additional UART, please open the UART as follows:



```
--- 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>;
```

Compile the kernel and download it. For more details, please refer to the compilation method in *KBA_QuecOpen_EC2X&AG35_Quick_Start*.
Download new firmware, if additional device nodes of */ dev / ttyHS1* are found, then UART1 can work.

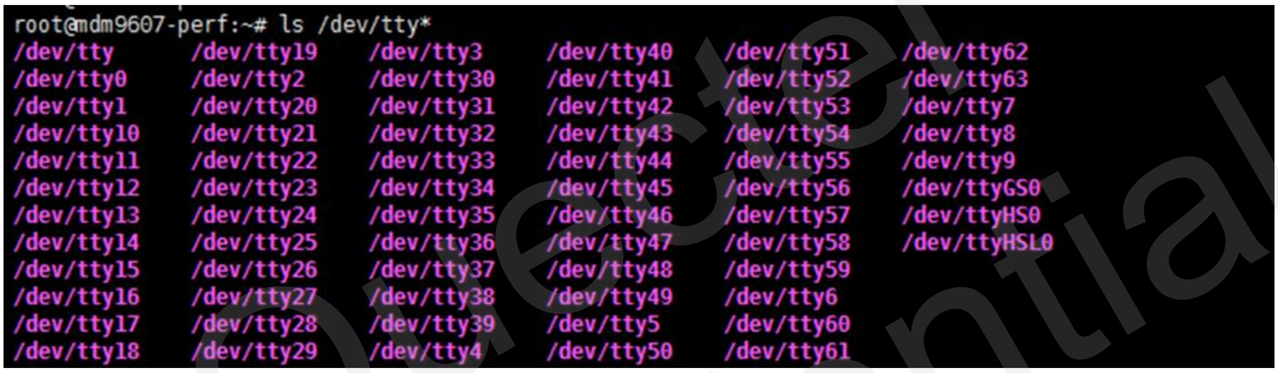
Quectel Confidential



```
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
```

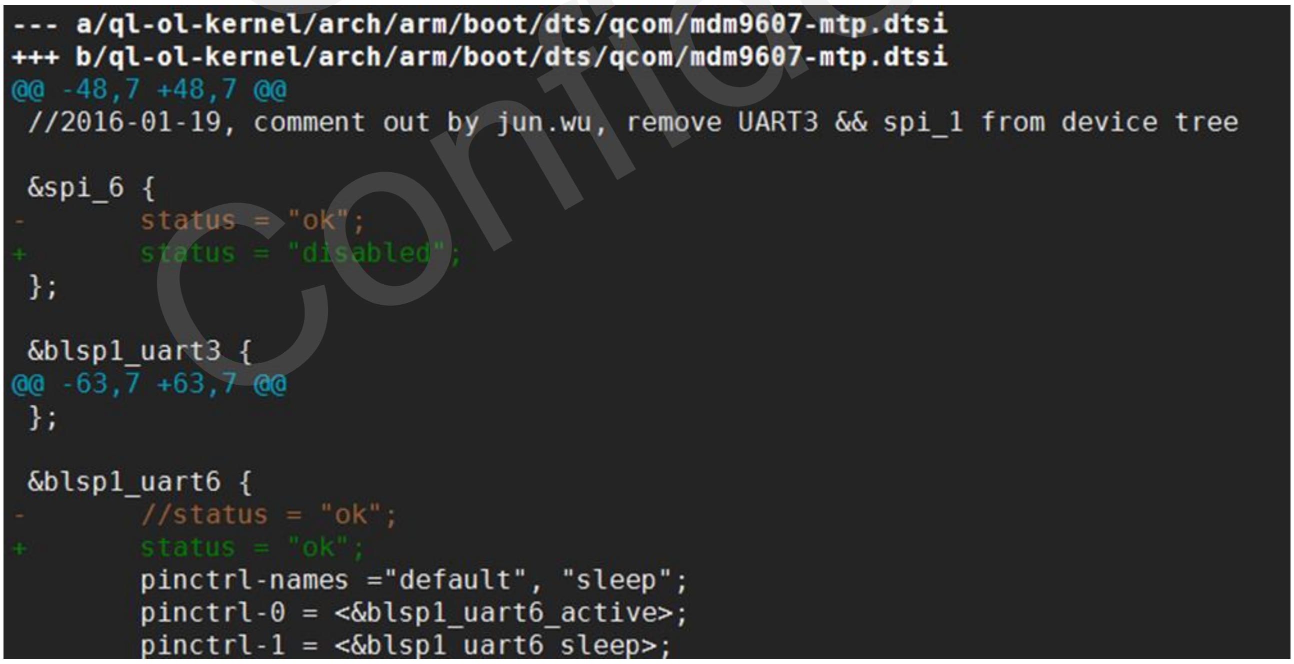
4.2.5. UART2 Configuration

UART2 is not enabled by default and UART device node are not displayed inside the module. PIN37, PIN38, PIN39 and PIN40 are configured as SPI interfaces.



```
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
```

If users need to add additional UART, open this UART as follows:

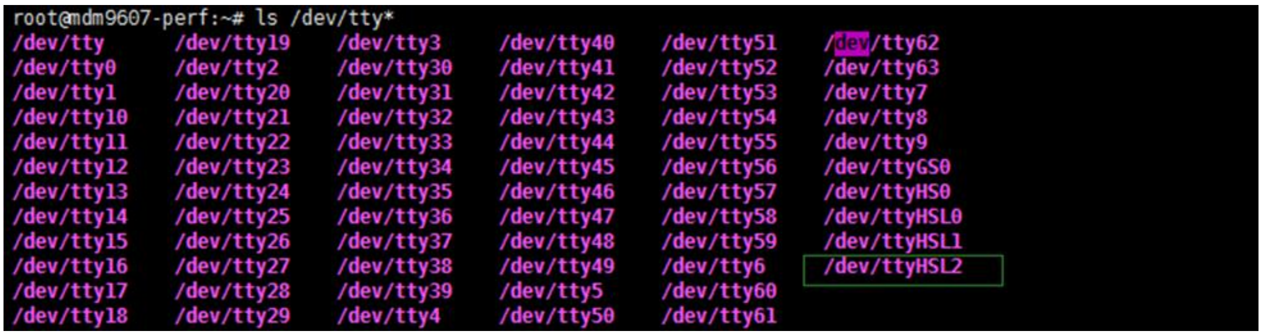


```
--- a/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
+++ b/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
@@ -48,7 +48,7 @@
 //2016-01-19, comment out by jun.wu, remove UART3 && spi_1 from device tree

 &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>;
```

Quectel Confidential

Compile the kernel and download it. For more details, please refer to the compilation method in *KBA_QuecOpen_EC2X&AG35_Quick_Start*.
Download new firmware, if additional device node of */dev/ttyHSL2* is found, then UART2 can work.

In addition, the UART number in the module is determined by the sequence of multiple UART nodes in the device tree made by the UART driver. If the above UART1 is not enabled, the UART2 node name will turn into ttyHSL1.



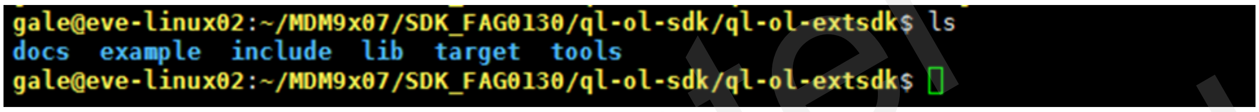
```
root@mdm9607-perf:~# ls /dev/tty*
/dev/tty      /dev/tty19    /dev/tty3     /dev/tty40    /dev/tty51    /dev/tty62
/dev/tty0     /dev/tty2     /dev/tty30    /dev/tty41    /dev/tty52    /dev/tty63
/dev/tty1     /dev/tty20    /dev/tty31    /dev/tty42    /dev/tty53    /dev/tty7
/dev/tty10    /dev/tty21    /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
```

Quectel
Confidential

5 QuecOpen Application Layer API

5.1. User Programming Instructions

A complete user programming interface is provided in QuecOpen project SDK.
Reference path: *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$
```

API interface library provided by Quectel is included under the *lib* directory in the above figure. The directory of *include* is the header file for all APIs. The directory of *example* provides function-based API usage example.

Here introduce UART related interfaces and reference examples.
UART application programming depends on library of *libql_peripheral.a.*
Head file: *ql_uart.h*.

5.2. UART API Introduction

Flow control enumeration: select flow control mode.

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

Parity bit enumeration: support none, odd parity and even parity.

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

Data bit enumeration: support 5, 6, 7 and 8.

Quectel
Confidential

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

Stop bit enumeration: support 1bit and 2bit.

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

Baud rate enumeration: baud rate supported by the module.

```
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;
```

UART properties structure:

```
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);*

Open a UART device file with the specified baud rate and flow control, and the default parity bit is PB_NONE, data bit DB_CS8, and stop bit SB_1. Use the interface of *int Ql_UART_SetDCB(int fd,*

Quectel Confidential

*ST_UARTDCB *dcb)* to modify these properties if neccessary.
Parameter: port: device name, such as /dev/ttyHS0
baudrate: baud rate, refer to enumeration Enum_BaudRat, such as B_9600, B_115200
flowCtrl: flow control, enumeration Enum_FlowCtrl
Return value: return file descriptor, otherwise return -1.

*int Ql_UART_SetDCB(int fd, ST_UARTDCB *dcb);*
Set UART properties, including flow control, parity bit, data bit, stop bit and baud rate.
Parameter: fd: device file descriptor
dcb: Filled UART properties structure
Return value: return 0, returns -1 when errors happened.

*int Ql_UART_GetDCB(int fd, ST_UARTDCB *dcb);*
Obtain the current UART properties, including flow control, parity bit, data bit, stop bit, and baud rate.
Parameter: fd: device file descriptor
dcb: UART properties structure
Return value: return 0, returns -1 when errors happened.

int Ql_UART_Read(int fd, char buf, unsigned int buf_len);*
Read the content of specified byte length from the UART to buf; return the read byte length
Parameter: fd: device file descriptor
buf: read data pointer
buf_len: read the length
Return value: return the read byte length.

int Ql_UART_Write(int fd, const char buf, unsigned int buf_len);*
Write the specified byte length data from buf to UART; return the written byte length.
Parameter: fd: device file descriptor
buf: write data pointer
buf_len: write the length
Return value: return the written byte length.

int Ql_UART_Close(int fd);
Close device file descriptor.

Advanced UART programming:
The above interfaces can fully meet the needs of customers.
If the user is an advanced Linux user and is very proficient in the characteristics of the UART, users can set the UART with the following interfaces.
int Ql_UART_IoCtl(int fd, unsigned int cmd, void pValue);*
Control UART device properties
Parameter: fd: device file descriptor
cmd: UART ioctl request, such as TCGETS, TCSETS, TIOCMGET, TIOCMSET, etc.
pValue: UART device properties pointer

Quectel Confidential

Return value: return 0 if succeed, otherwise return -1.

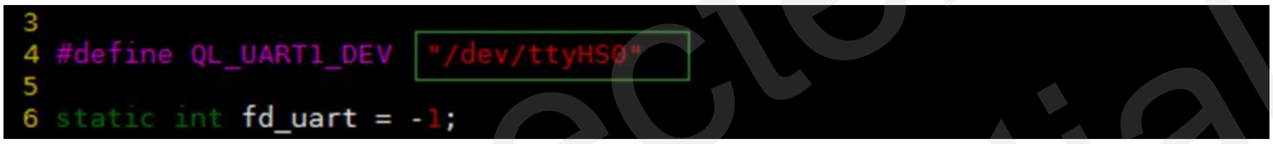
Users can monitor the device file descriptor of uart_fd to implement asynchronous notification to read UART data.
Please refer to: *ql-ol-extsdk/example/uart.*

Quectel Confidential

6 Verification for UART Function Test

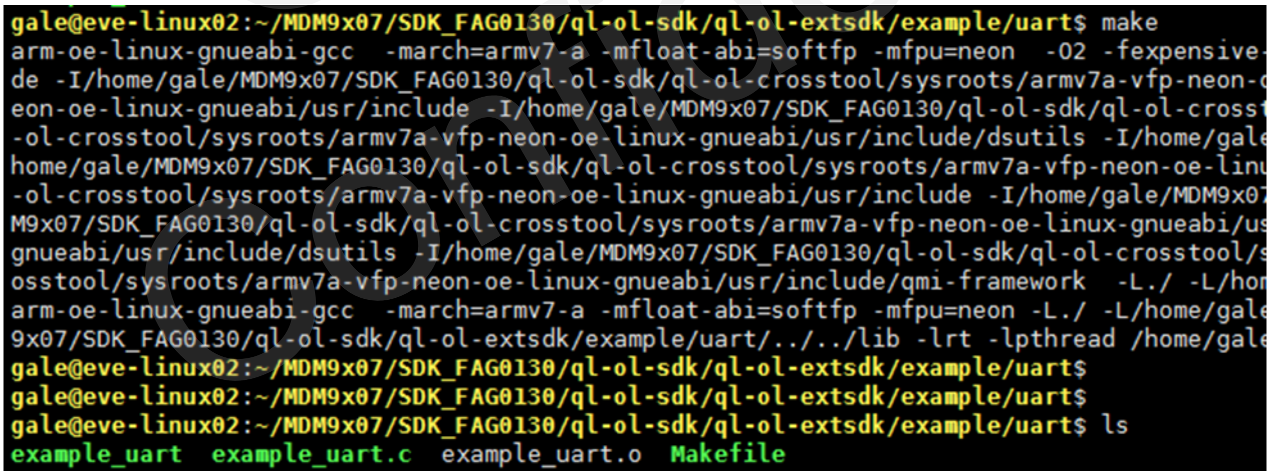
6.1. Introduction and Compilation of Example

Here is an example of main UART, namely */dev/ttyHS0.*
In the example, the main thread writes data to the UART every second, and users can open the COM port of the host to receive data; at the same time, the child thread is monitoring RX if there is data coming in. Users send data to RX from the COM port of the host, the main UART will receive and print it out.



```

#define QL_UART1_DEV  "/dev/ttyHS0"

static int fd_uart = -1;
```

Enter the directory of *ql-ol-sdk/ql-ol-extsdk/example/uart*, and *make* generates executable program of *example_uart.* The premise of compiling is that the initialization of the cross-compilation environment has been completed.
Source *ql-ol-crosstool/ql-ol-crosstool-env-init*

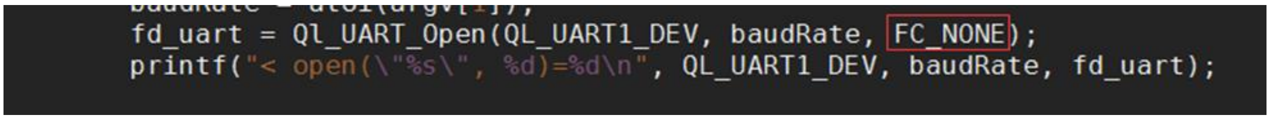


```
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk/example/uart$ make
arm-oe-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=softfp -mfpu=neon  -O2 -fexpensive-
de -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-o
eon-oe-linux-gnueabi/usr/include -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosst
-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
```

Quectel Confidential

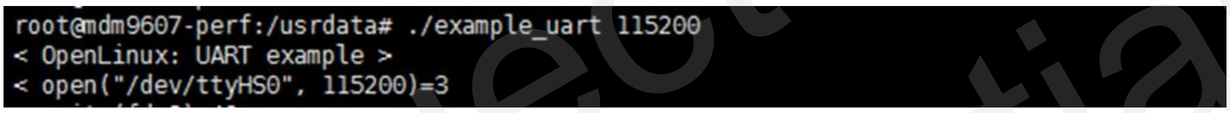
6.2. Function Test

6.2.1. Disabling Flow Control



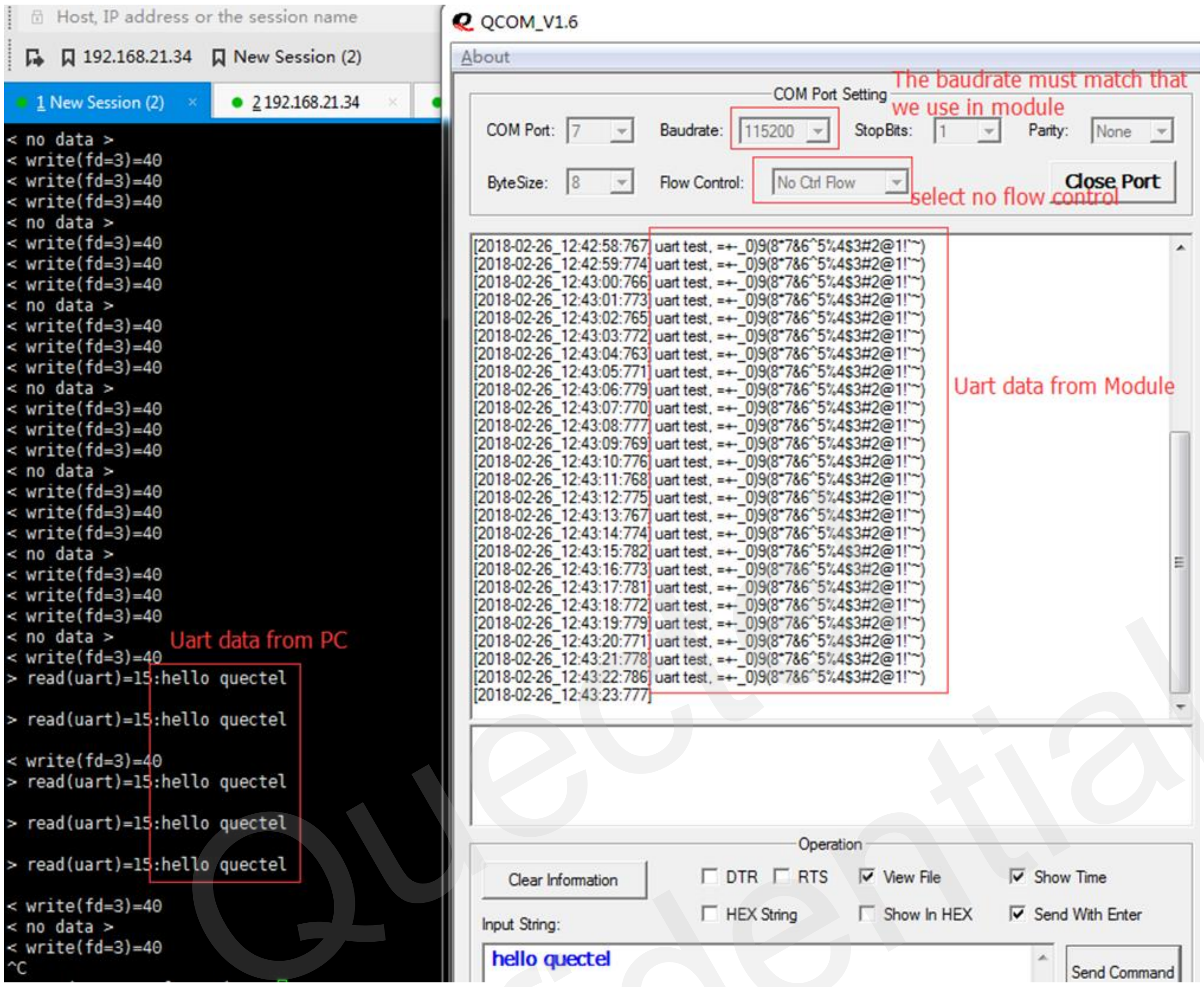
```
    fd_uart = Ql_UART_Open(QL_UART1_DEV, baudRate, FC_NONE);
    printf("< open(\"%s\", %d)=%d\n", QL_UART1_DEV, baudRate, fd_uart);
```

1. Upload the compilation example_uart to the module via adb push<the path of example_uart in the host computer>, < internal path of the module, such as /usrdata> or UART protocol, RZ.
2. If users need to use OPEN_EVB, it is necessary to connect the J0202's main UART pin header with a jumper cap.
   GPIO_0 connects to MAIN_TXD;
   GPIO_1 connects to MAIN_RXD;
   Then connect to the hardware path.
3. Execute example_uart 115200, and open the corresponding COM port with corresponding baud rate and no flow control on the host as follows.



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

Quectel Confidential



```
< 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
< 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
> read(uart)=15:hello quectel

> read(uart)=15:hello quectel

< write(fd=3)=40
> read(uart)=15:hello quectel

> read(uart)=15:hello quectel

> read(uart)=15:hello quectel

< write(fd=3)=40
< no data >
< write(fd=3)=40
^C
```

COM Port: 7 Baudrate: 115200 StopBits: 1 Parity: None
ByteSize: 8 Flow Control: No Ctrl Flow

```
[2018-02-26_12:42:58:767] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:42:59:774] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:00:766] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:01:773] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:02:765] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:03:772] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:04:763] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:05:771] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:06:779] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:07:770] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:08:777] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:09:769] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:10:776] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:11:768] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:12:775] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:13:767] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:14:774] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:15:782] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:16:773] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:17:781] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:18:772] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:19:779] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:20:771] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:21:778] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:22:786] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_12:43:23:777]
```

☐ DTR ☐ RTS ☑ View File ☑ Show Time
☐ HEX String ☐ Show In HEX ☑ Send With Enter

Input String: hello quectel

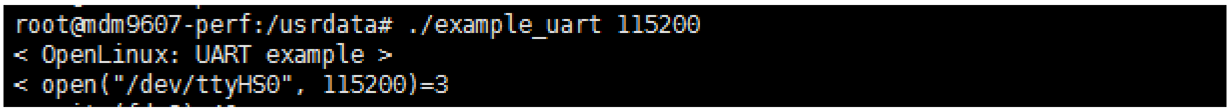
6.2.2. Enabling Hardware Flow Control

Modify example to enable hardware flow control.

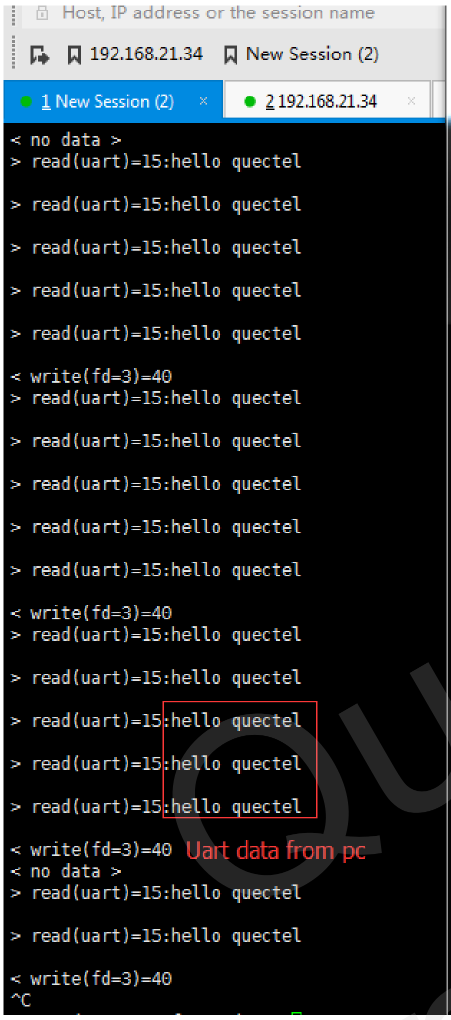
```
    fd_uart = Ql_UART_Open(QL_UART1_DEV, baudRate, FC_RTSCTS);
    printf("< open(\"%s\", %d)=%d\n", QL_UART1_DEV, baudRate, fd_uart);
```

1. Upload the compilation example_uart to the module via adb push<the path of example_uart in the host >, < internal path of the module, such as /usrdata> or UART protocol, RZ.
2. If users need to use OPEN_EVB, it is necessary to connect the J0202's main UART pin header with a jumper cap.
   GPIO_0 connects to MAIN_TXD;
   GPIO_1 connects to MAIN_RXD;
   GPIO_2 connects to MAIN_RTS;
   GPIO_3 connects to MAIN_CTS;
   Then connect to the hardware path.
3. Execute example_uart 115200, and open the corresponding COM port with corresponding baud rate and hardware flow control on the host as follows.

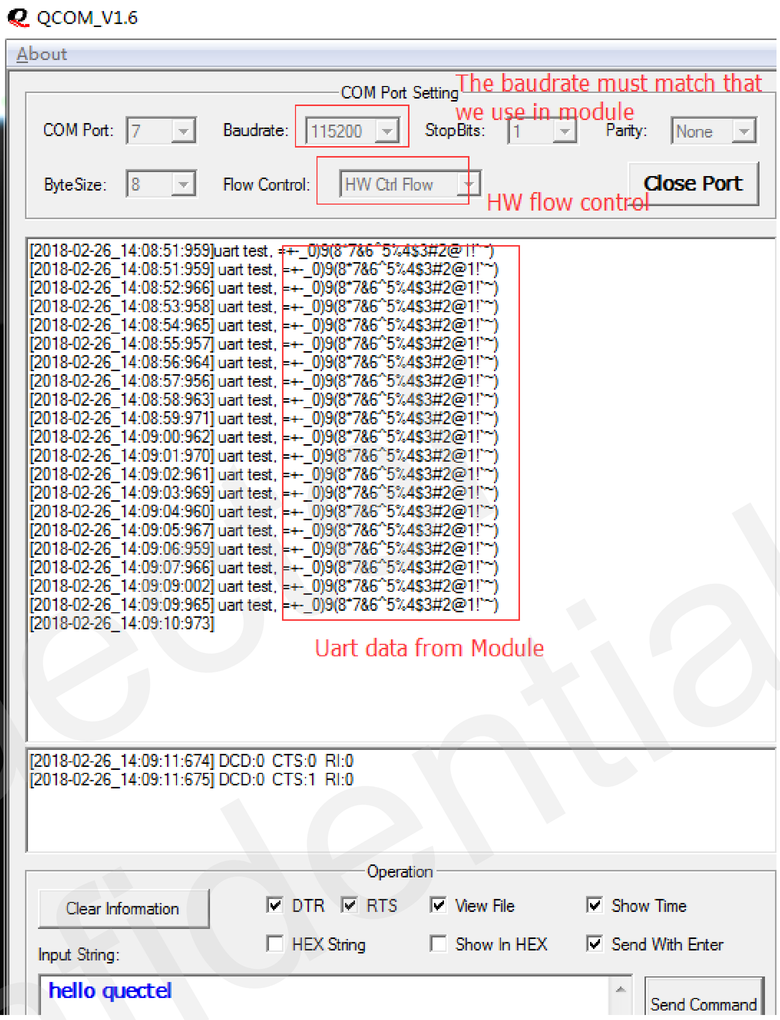
Quectel Confidential



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



```
< no data >
> 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
> 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
> 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

< write(fd=3)=40
^C
```



COM Port: 7 Baudrate: 115200 StopBits: 1 Parity: None
ByteSize: 8 Flow Control: HW Ctrl Flow

```
[2018-02-26_14:08:51:959]uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:51:959] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:52:966] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:53:958] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:54:965] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:55:957] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:56:964] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:57:956] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:58:963] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:08:59:971] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:00:962] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:01:970] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:02:961] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:03:969] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:04:960] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:05:967] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:06:959] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:07:966] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:09:002] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:09:965] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_14:09:10:973]
```

```
[2018-02-26_14:09:11:674] DCD:0 CTS:0 RI:0
[2018-02-26_14:09:11:675] DCD:0 CTS:1 RI:0
```

☑ DTR ☑ RTS ☑ View File ☑ Show Time
☐ HEX String ☐ Show In HEX ☑ Send With Enter

Input String: hello quectel

6.2.3. Enabling Software Flow Control

6.2.3.1. Description of Software Flow Control XON/XOFF Character

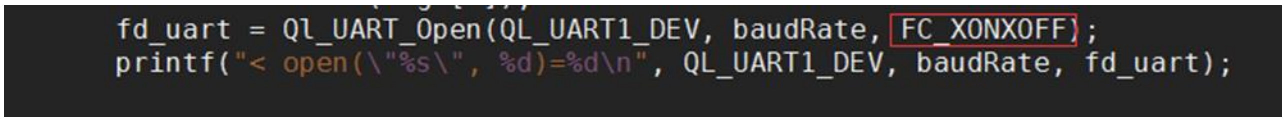
XOFF/XON representations in ASCII

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

Quectel Confidential

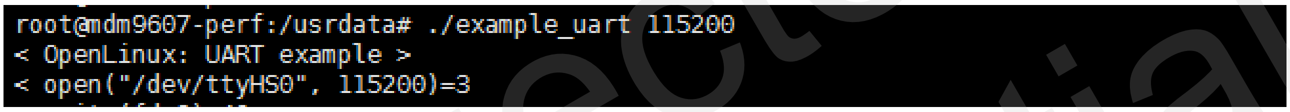
6.2.3.2. Software Flow Control Test

Modify example to enable hardware flow control.



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

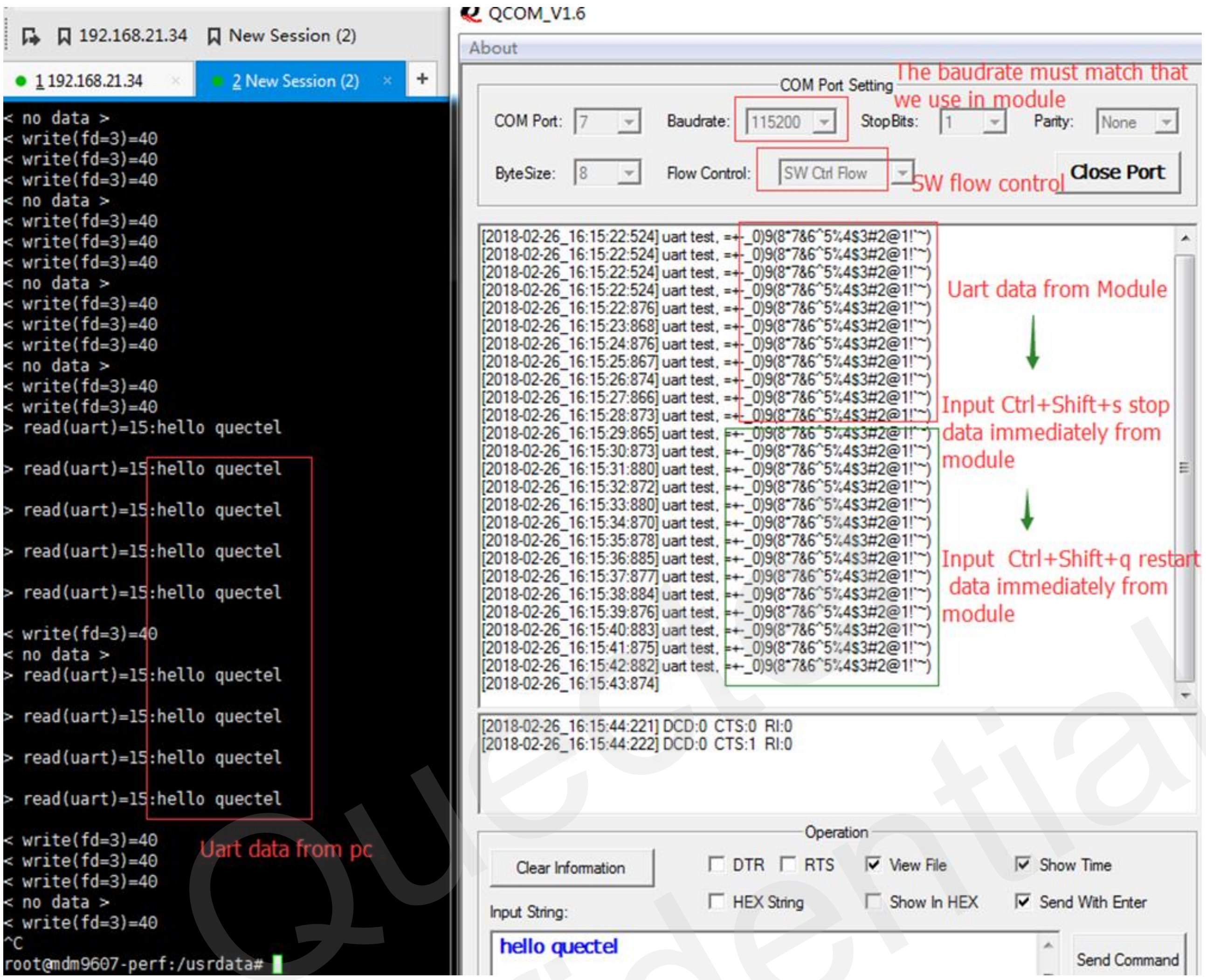
1. Upload the compilation example_uart to the module via adb push<the path of example_uart in the host computer>, < internal path of the module, such as /usrdata> or UART protocol, rz.
2. If users need to use OPEN_EVB, it is necessary to connect the J0202's main UART pin header with a jumper cap.
   GPIO_0 connects to MAIN_TXD;
   GPIO_1 connects to MAIN_RXD;
   Then connect to the hardware path.
3. Execute example_uart 115200, and open the corresponding COM port with corresponding baud rate and hardware flow control on the host as follows.



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

Type Ctrl+Shift+S on the keyboard in the UART software of the host, or send hexadecimal 0x13 when Enabling software flow control to transfer data, the module side will stop data transmission immediately. Resume data transmission by Ctrl+Shift+Q or sending 0x11 to verify that the software flow control is normal.

Quectel Confidential



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#
```

QCOM_V1.6

COM Port Setting: COM Port: 7; Baudrate: 115200; StopBits: 1; Parity: None; ByteSize: 8; Flow Control: SW Ctrl Flow

```
[2018-02-26_16:15:22:524] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:22:524] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:22:524] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:22:524] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:22:876] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:23:868] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:24:876] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:25:867] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:26:874] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:27:866] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:28:873] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:29:865] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:30:873] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:31:880] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:32:872] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:33:880] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:34:870] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:35:878] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:36:885] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:37:877] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:38:884] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:39:876] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:40:883] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:41:875] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:42:882] uart test, =+-_0)9(8*7&6^5%4$3#2@1!`~)
[2018-02-26_16:15:43:874]
```

```
[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: [ ] DTR [ ] RTS [✓ View File] [✓ Show Time] [ ] HEX String [ ] Show In HEX [✓ Send With Enter]

Input String: hello quectel

About
Close Port
Clear Information
Send Command
The baudrate must match that we use in module
SW flow control
Uart data from Module
Input Ctrl+Shift+s stop data immediately from module
Input Ctrl+Shift+q restart data immediately from module
Uart data from pc

Quectel Confidential